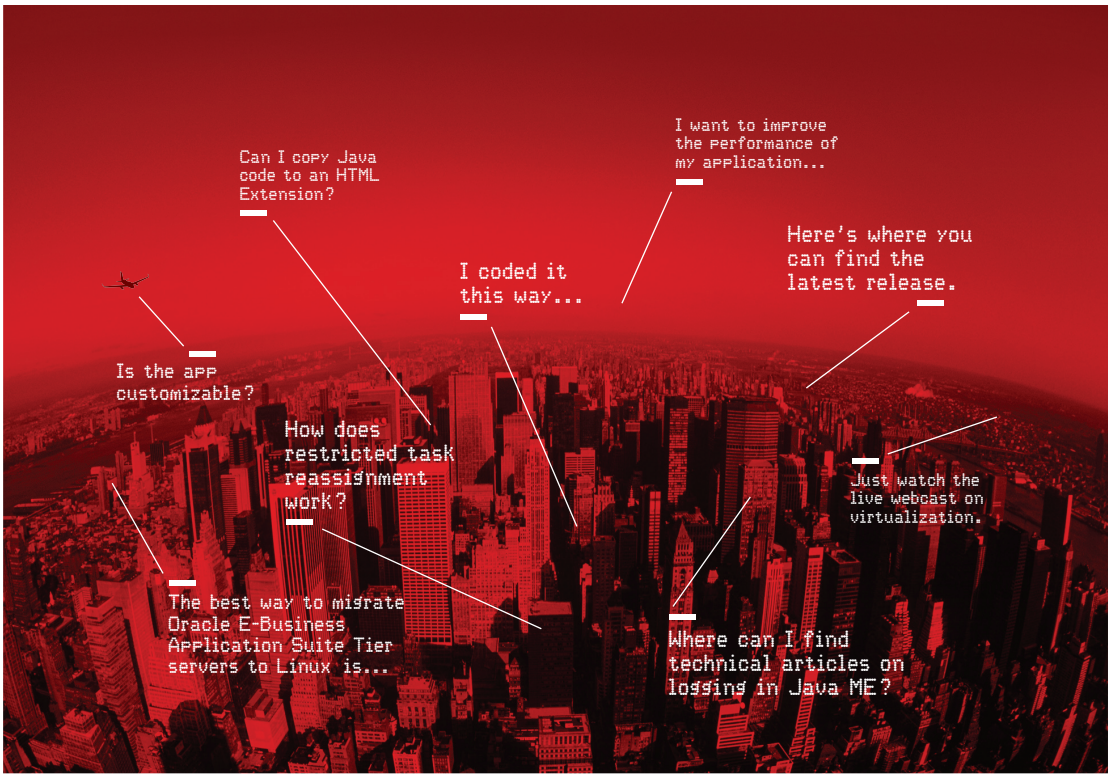
Hands on Lab Manual

Oracle WebLogic Server 12 *c*

ORACLE
TECHNOLOGY NETWORK

http://www.oracle.com/technetwork



OTN Virtual Developer Day 2012 - Oracle WebLogic Server 12*c*

Steve Button, Pieter Humphrey, Greg Stachnick, Arun Gupta
1/13/12 1:00 PM
Version 1.3

Introduction

Part 1: Developing Java EE 6 Applications with Oracle Enterprise Pack for Eclipse and WebLogic Server 12*c*

Overview

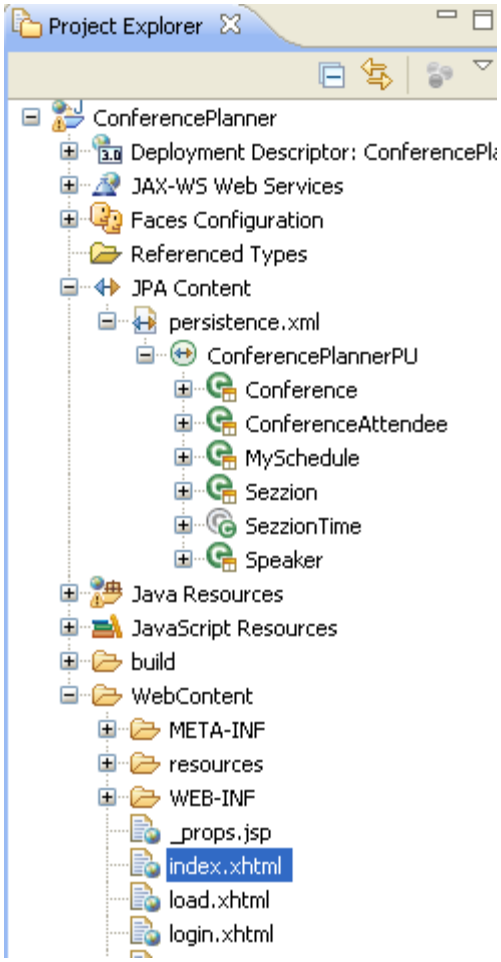
In Part 1, you will build and deploy the application that you'll be working with in Maven to WebLogic 12*c* from Eclipse, test it in a browser, change the data source configuration, and work with the Server console before diving into Maven at the command line in Parts 2 and 3.

Build and Deploy Application with Eclipse

The Conference Planner application has been made available for you to work with Oracle Enterprise Pack for Eclipse (OEPE) and to explore, change and experiment with as you see fit.

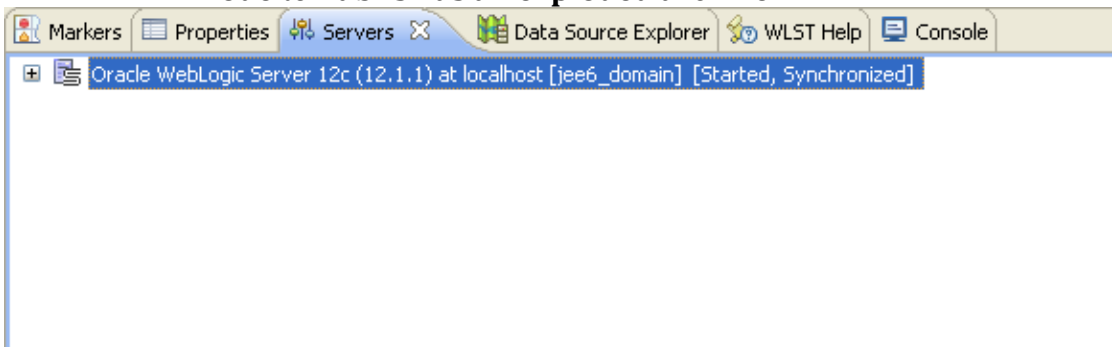
1.2 Explore the Conference Planner Project using the Project Explorer window.

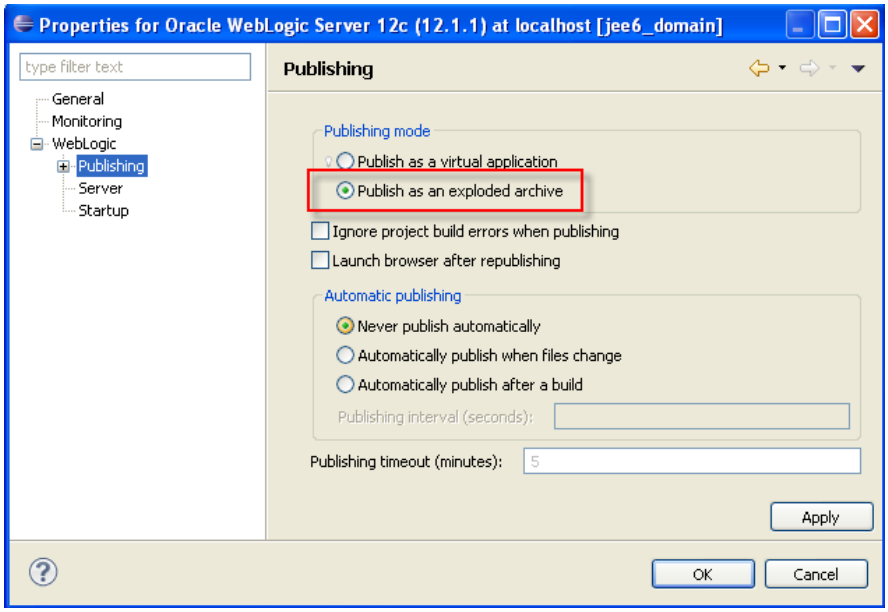
The ConferencePlanner web application should appear in the Project Explorer with no build errors. If build errors exist, try rebuilding the project with **Project** menu > **Clean**.



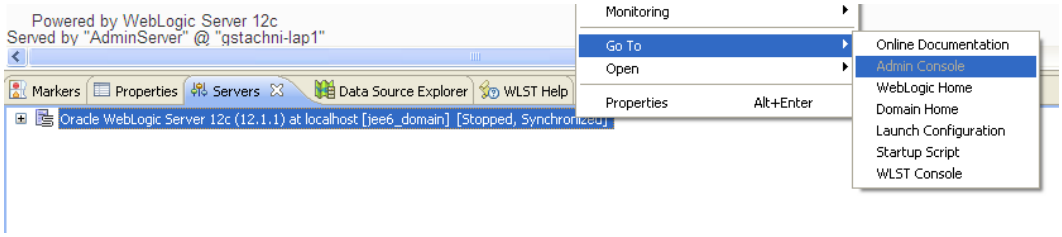
By default, OEPE deploys and runs your projects directly out of the workspace by taking advantage of WebLogic Server's split source feature. This provides better performance for iterative development and debugging. For this exercise, change the default deployment strategy to use a packaged archive.

- Right click the server configuration and select **Properties**
- Under the **WebLogic > Publishing** section, change the **Publishing Mode** to **Publish as an exploded archive**





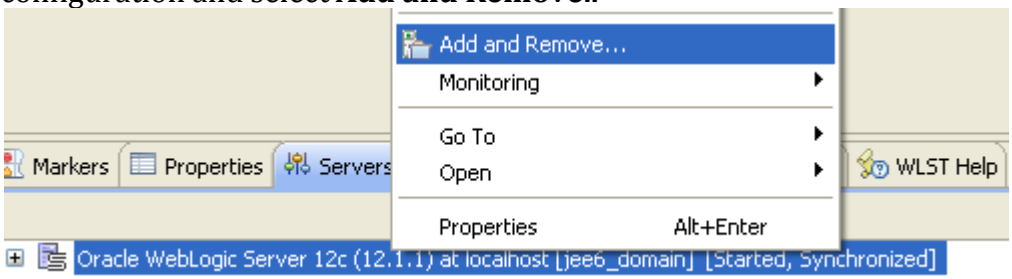
OEPE makes it easy to access numerous configurations of WebLogic Server 12*c*. To quickly access the console, right-click the WebLogic Server configuration and select **Start** and then select **Go To > Admin Console** once the server has started.



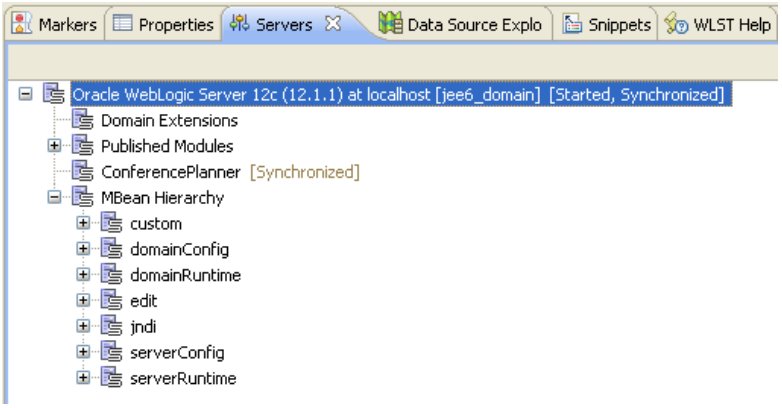
1.3 Run the Conference Planner on WebLogic Server from OEPE.

With the Oracle WebLogic Server resource created, the Conference Planner can be run using it directly from Eclipse.

To deploy the ConferencePlanner, right click the WebLogic Server 12.1.1 configuration and select **Add and Remove..**

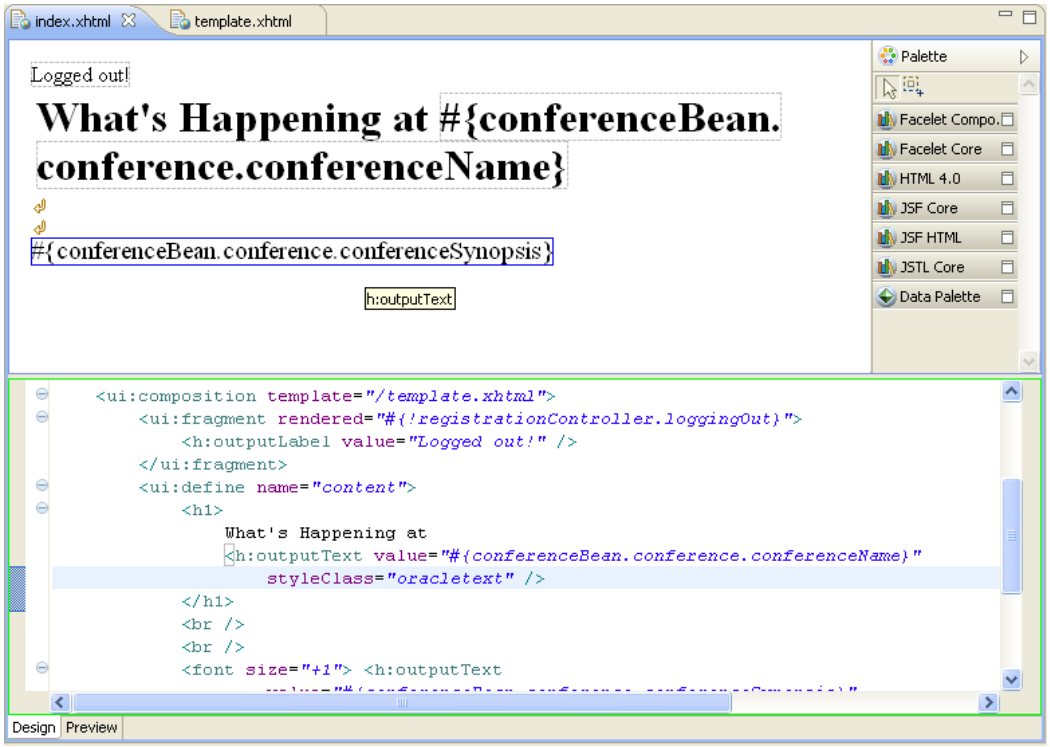


Add the ConferencePlanner project to configure with the server and select **Finish**.
In the **Servers** view, right click the WebLogic Server configuration and select **Publish**.
WebLogic Server should start successfully. Expand the server configuration to review the available services including **Published Modules**, **Deployed Applications**, and the **MBean Hierarchy**.



1.4 Test the application.

The starting page for Conference Planner is **index.xhtml** found under the **WebContent** directory. Double click **index.xhtml** to open in the **Web Page Editor** and review the source.



To run the page, right click **index.xhtml** and **select Run As > Run On Server.** Accept the default server settings and click **Finish**. The page should load in the integrated browser as shown below.

1.5 Change to @DataSourceDefinition and redeploy.

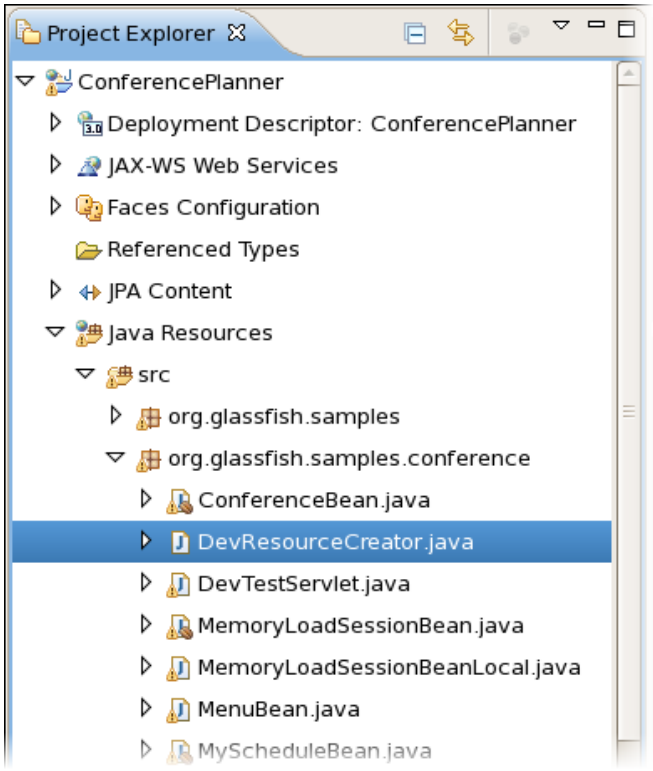
The default state of the application is to use an external datasource with a JNDI name of "jdbc/demo".

This is specified as the <jta-data-source> element in the JPA persistence.xml file.

Java EE 6 has provided an alternate mechanism of defining a datasource using an annotation @DataSourceDefinition as a developer convenience. This annotation

is placed on a Java EE component with the required connection settings, and the server will create a corresponding DataSource at runtime. An EJB is provided in the project, which has this annotation already defined on it.

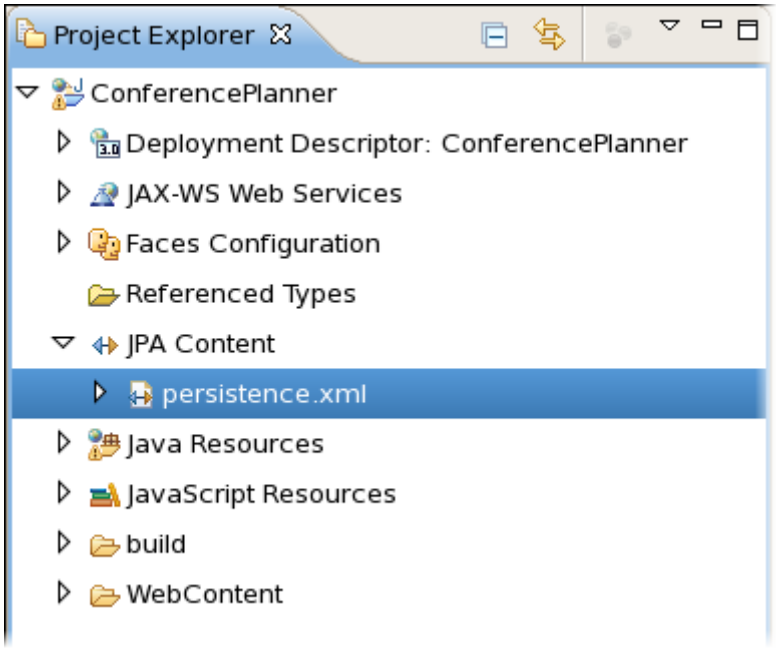
Locate and open the DevResourceCreator.java file for editing.



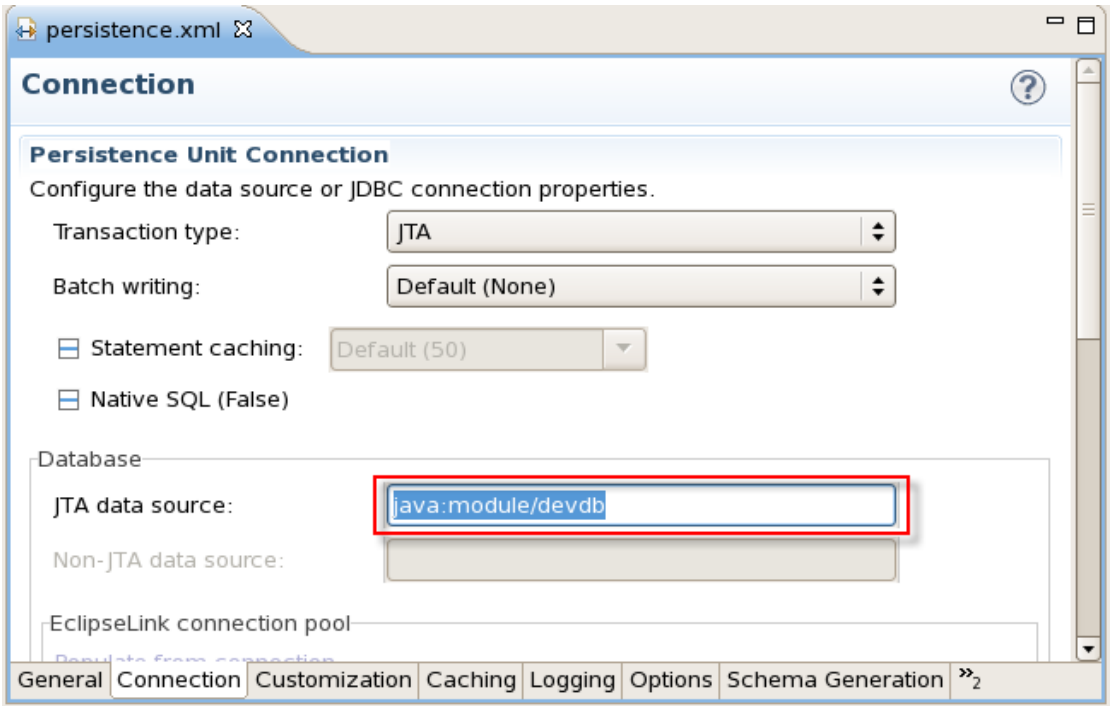
Review the @DataSourceDefinition, and uncomment the settings for Derby. Note that there is also an example annotation for Oracle XE (although Oracle XE is not installed on the VirtualBox image). You can leave the Oracle setting uncommented as well.

```
@DataSourceDefinition(
   name="java:module/devdb",
   description="@DataSourceDefinition for dev testing",
   minPoolSize = 0,
   initialPoolSize = 0,
   className = "org.apache.derby.jdbc.ClientXADataSource",
   user = "app",
   password = "app",
   url="jdbc:derby://localhost:1527/demo;databaseName=demo"
)
```

Locate the persistence.xml file and open it for editing.



Change the jta-data-source element to a value of "java:module/devdb" to use the @DataSourceDefinition defined DataSource.



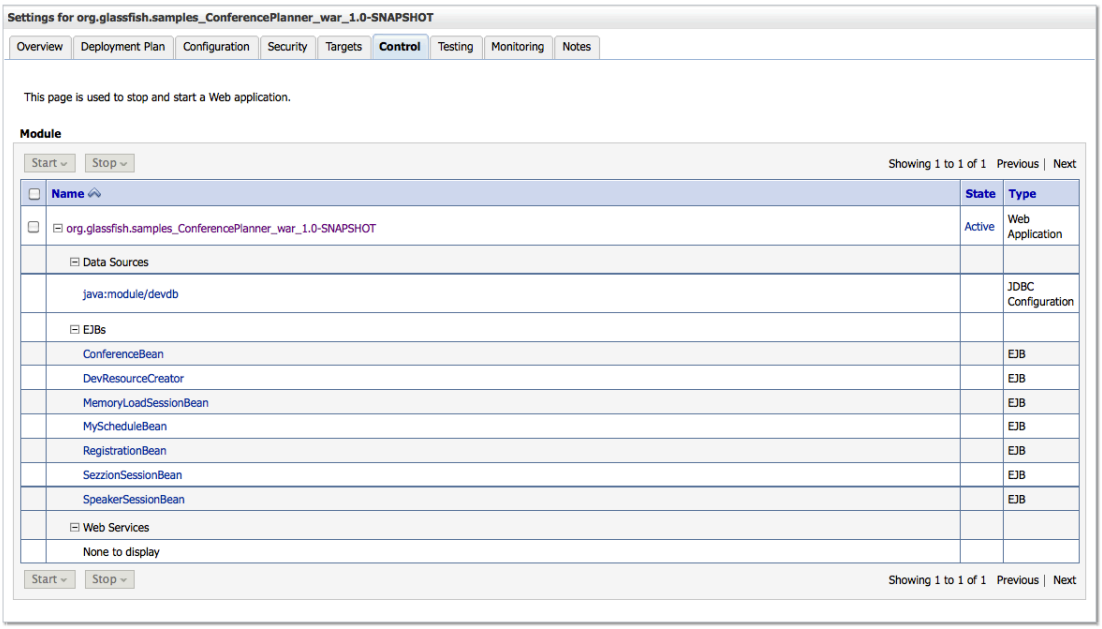
Save the file.

Save all changes, right click the WebLogic Server 12*c* configuration in the **Servers** tab, and select **Publish** to redeploy the application.

After you have deployed the application, using the WebLogic Server Administration Console, you will see the @DataSourceDefinition defined Data Source as a module of the Conference Planner application. You can access the

console by right clicking on the entry in the servers tab and choosing Go to -> Admin Console as described above in 1.2.

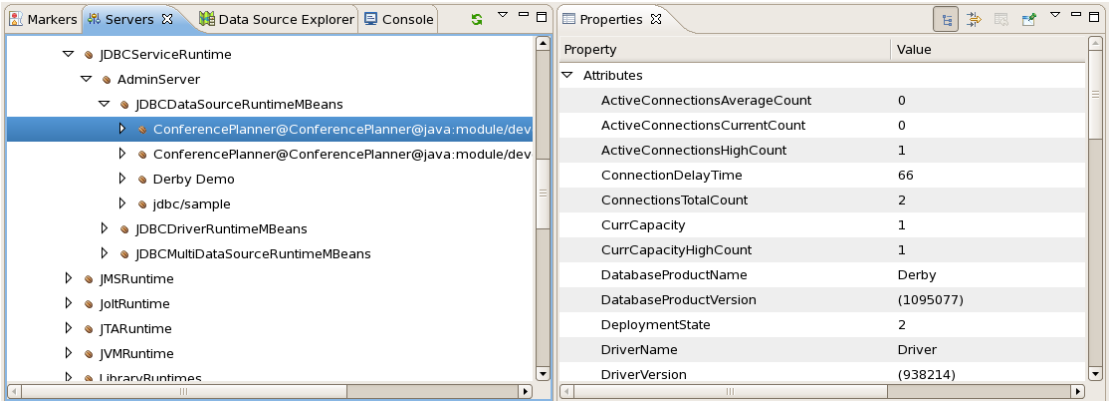
To view its configuration: click the deployments link in the domain structure box, and then on the left side of the console, click on the link for the conference planner web application deployment. Then, from the control tab, click on java:/module/devdb and select the Configuration tab. (In Linux, you can get rid of the console popup by foregrounding the embedded browser).



Additional information about this data source can be found in the MBean tree hierarchy. In the **Servers** view in Eclipse, expand the **MBean Hierarchy** node to browse the MBeans for this server instance.

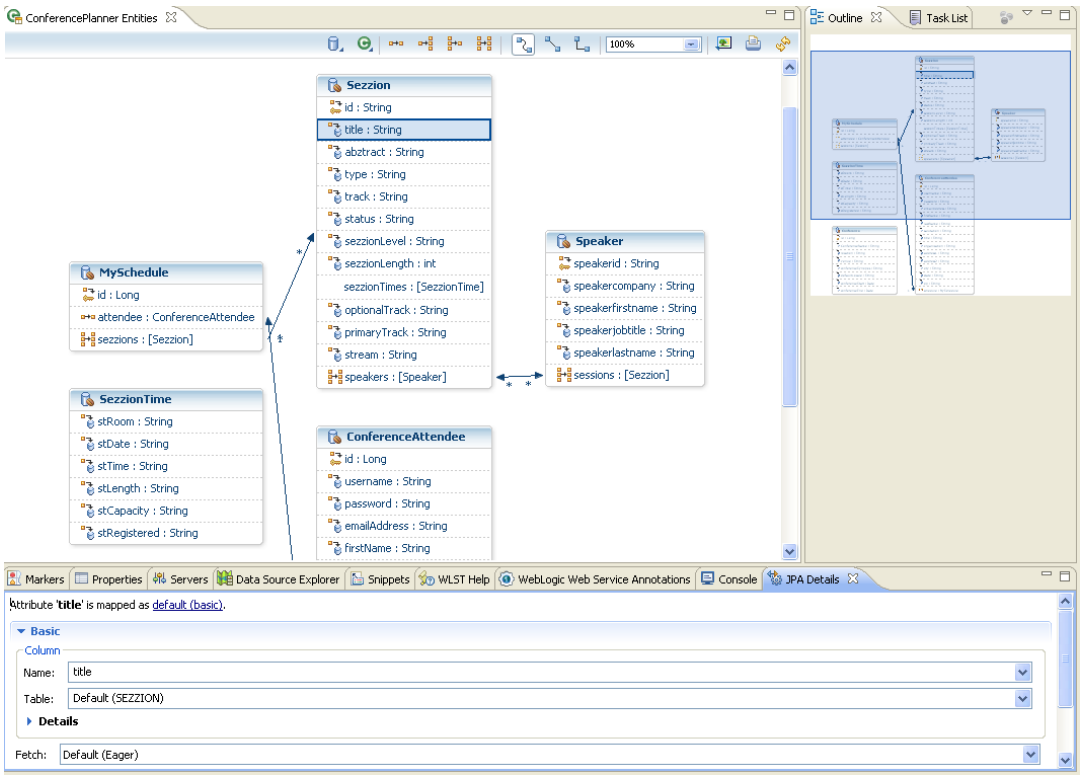
serverRuntime:/JDBCServiceRuntime/AdminServer/JDBCDataSourceRuntimeMBeans/

The Properties sheet displays the specific configuration details for this data source.



1.6 Review JPA Entity information.

Oracle Enterprise Pack for Eclipse provides a rich set of tools for Java Persistence development. To view the **Entity Diagram** for the Conference Planner application, right click the project and select **JPA Tools > Show in Entity Editor.**



The **Entity Editor** displays all entity classes and their relationships in a single view. Right clicking a node will provide navigation options such as opening the Java Source. To edit entities from the **Entity Editor**, open the **JPA Details** view. Click the **Window** menu > **Show View > Other**, and then select **JPA > JPA Details view**. The **JPA Details** view displays the properties for the node selection in either the Entity Editor or the Java Source.

1.7 Break Free! Explore on your own.

You have successfully built, packaged and deployed the application. You are now in a position to make further changes to the application to test out new APIs or features of Java EE 6, with the knowledge that you can successfully build and deploy the application to WebLogic Server 12*c*.

Ideas?

- Find the template.xhtml and modify the template to see the effect.
- Change the ClusterInstanceBean to CDI with an @Produces to generate server specific instances.
- Change the validation constraints definitions on the ConferenceAttendee @Entity to change how is validated.

- Add a Servlet to the application using the @WebServlet annotation, inject a CDI bean of some sort.
- Use the CDI event model to send an event whenever a new user is registered and handle the event somewhere with an @Observes.
- Create a new @Entity to keep track of registration events and store them in the database via a new JPA @Entity Audit.
- Add a new JSF page to the application, using the same JSF template, which displays the Audit entries in a table. Add the new page to the menu bar.
- Expose a JAX-RS interface for the SezzionSessionBean EJB and test it out. Can you return the Session set in both JSON and XML formats?

Summary

In Part 1, you built and deployed the conference planner application and made changes to the datasource configuration. Now it's time to dive into Maven at the command line, Hudson and WebLogic Server 12*c*.

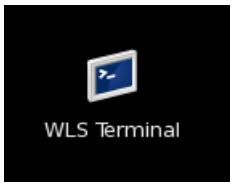
Part 2: Using the WebLogic Maven Plugin to Automate the Build, Deploy and Test Cycle

Overview

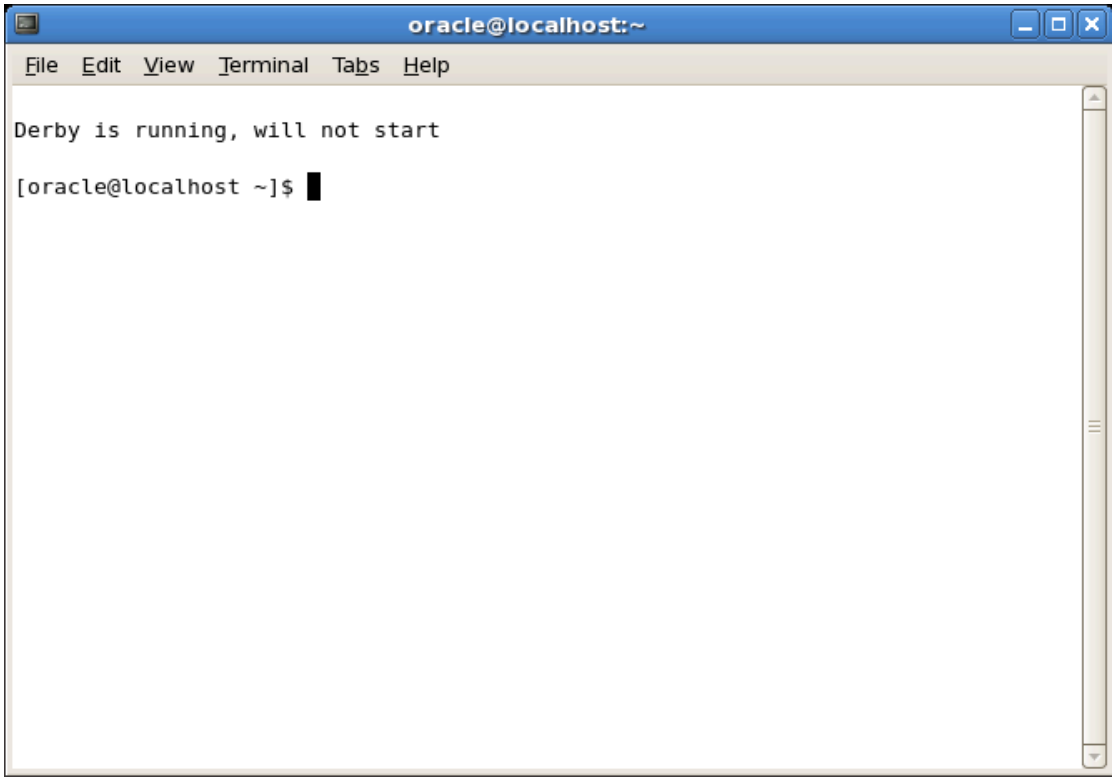
In part 1 of this hands-on lab, you used a native OEPE project to build and deploy the Java EE 6 Conference Planner application to a WebLogic Server 12c domain, and tested it manually.

In part 2 of this hand-on lab, you will now work with a maven version of the same Conference Planner application. Using maven you will be building and packaging the application from the command line. You will then use the new wls-maven-plugin provided in 12*c* to execute a fully automated testing lifecycle of the application covering the initial installation of WebLogic Server 12*c*; the creation of a new domain; the starting of the server; construction of a data-source; the deployment of the application and the execution of a simple integration test to validate the correct working operation of the application.

1. Open a new WLS Terminal using the desktop icon



This will create a new terminal for you to use to complete this section of the hands-on lab.



Note: the "Derby is running, will not start" message will appear if the local Derby database has been previously started by running a terminal. It can be safely ignored.

2. Check the Conference Planner application out from the local subversion repository

```
$ cd ~/labs/Dev_labs_2012/WLS-Maven

$ svn co svn://localhost/otnvdd/conference-planner

A    conference-planner/trunk
A    conference-planner/trunk/src
A    conference-planner/trunk/src/test
A    conference-planner/trunk/src/test/java
...
```

3. Build the application using maven

The application can be built using the standard maven lifecycle package, which will build and package the application into the specified target war file.

```
$ cd conference-planner/trunk

$ mvn package

[INFO] Scanning for projects...
[INFO]
```

```
[INFO] ------------------------------------------------------------------
----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------
----------

...

[INFO] --- maven-war-plugin:2.1.1:war (default-war) @
ConferencePlanner ---
[INFO] Packaging webapp
[INFO] Assembling webapp [ConferencePlanner] in
[/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/ConferencePlanner-1.0-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources
[/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/src/main/webapp]
[INFO] Webapp assembled in [86 msecs]
[INFO] Building war: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/ConferencePlanner.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO] ------------------------------------------------------------------
----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
----------
[INFO] Total time: 3.876s
[INFO] Finished at: Mon Dec 19 18:56:02 PST 2011
[INFO] Final Memory: 13M/31M
[INFO] ------------------------------------------------------------------
----------
```

Check that the war file exists in the target directory.

```
$ ls target

classes                              ConferencePlanner.war
generated-sources                    test-classes
ConferencePlanner-1.0-SNAPSHOT       endorsed
maven-archiver
```

4. Install the wls-maven-plugin into the local maven repository

Before using the wls-maven-plugin, it first must be installed into the local maven repository. To do this, you use the mvn:install and mvn:install:install-file goals.

The libraries you need are in the labs/Dev_labs_2012/WLS-Maven/binaries directory. Note: there is a pom.xml in the binaries directory.

Note: you need to execute BOTH goals shown below in bold type.

First execute the simple mvn:install goal to install the plugin meta-data into the repository from the local pom.xml file.

```
$ cd ~/labs/Dev_labs_2012/WLS-Maven/binaries

$ ls

pom.xml  wls1211_dev.zip  wls-maven-plugin-12.1.1.0.jar

$ mvn install

INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building WebLogic Server Maven Plugin 12.1.1.0
[INFO] ------------------------------------------------------------
-----------
[INFO]
...
INFO] --- maven-install-plugin:2.3.1:install (default-install) @
wls-maven-plugin ---
[INFO] Installing /home/oracle/labs/Dev_labs_2012/WLS-
Maven/binaries/target/wls-maven-plugin-12.1.1.0.jar to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-maven-
plugin/12.1.1.0/wls-maven-plugin-12.1.1.0.jar
[INFO] Installing /home/oracle/labs/Dev_labs_2012/WLS-
Maven/binaries/pom.xml to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-maven-
plugin/12.1.1.0/wls-maven-plugin-12.1.1.0.pom
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 8.576s
[INFO] Finished at: Thu Dec 22 18:48:59 PST 2011
[INFO] Final Memory: 4M/15M
[INFO] ------------------------------------------------------------
-----------
```

Now execute the **mvn install:install-file** goal to actually upload the plugin to the repository.

```
$ mvn install:install-file -Dfile=wls-maven-plugin-12.1.1.0.jar -
DpomFile=pom.xml

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building WebLogic Server Maven Plugin 12.1.1.0
[INFO] ------------------------------------------------------------
-----------
[INFO]
```

```
[INFO] --- maven-install-plugin:2.3.1:install-file (default-cli) @
wls-maven-plugin ---
[INFO] Installing /home/oracle/labs/Dev_labs_2012/WLS-
Maven/binaries/wls-maven-plugin-12.1.1.0.jar to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-maven-
plugin/12.1.1.0/wls-maven-plugin-12.1.1.0.jar
[INFO] Installing /home/oracle/labs/Dev_labs_2012/WLS-
Maven/binaries/pom.xml to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-maven-
plugin/12.1.1.0/wls-maven-plugin-12.1.1.0.pom
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 0.527s
[INFO] Finished at: Mon Dec 19 19:35:31 PST 2011
[INFO] Final Memory: 2M/15M
[INFO] ------------------------------------------------------------
-----------
```

And finally, test the wls-maven-plugin was completed successfully using the wls:help goal.

```
$ mvn wls:help

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building WebLogic Server Maven Plugin 12.1.1.0
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:help (default-cli) @ wls-maven-
plugin ---
[INFO]
WebLogic Server Maven Plugin
The following goals are supported by wls-maven-plugin:

appc:
  The appc compiler generates and compiles the classes needed to
deploy
  EJBs and JSPs to WebLogic Server. It also validates the deployment
  descriptors for compliance with the current specifications at both
the
  individual module level and the application level.

deploy:
  To deploy a weblogic server application (supports all formats
WAR,JAR
  etc..)

create-domain:
  Create a domain for WebLogic Server using the default domain
template.
  For more complex domain creation use the WLST goal.
```

```
...

wlst:
  WLST wrapper for Maven


For detailed help on a goal, use -Dgoal=<goal-name> -Ddetail=true
options.
[INFO] ------------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
-----------
[INFO] Total time: 0.580s
[INFO] Finished at: Mon Dec 19 19:36:21 PST 2011
[INFO] Final Memory: 4M/15M
[INFO] ------------------------------------------------------------------
----------
```

5. Install the WebLogic Server developer zip distribution into the local maven repository

The wls-maven-plugin has a wls:install goal that can be used to perform an installation of a WLS distribution on the local server. The install goal can be configured to with the distribution to install in one of 3 ways: a local file system reference; a HTTP reference; a maven artifact reference.

For the purposes of this lab, you will be installing a wls1211_dev.zip from the local maven repository. This allows the same distribution to be used seamlessly across any maven project or execution environment on the local server.

Note: it's entirely possible to have the wls1211_dev.zip stored in a shared corporate level maven repository, from where it could be automatically fetched using the standard maven remote repository model.

For the purposes of this hands-on lab, we will be installing the wls1211_dev.zip into the local maven repository with the following coordinates:

groupId: **com.oracle.weblogic**
artifactId: **wls-dev**
version: **12.1.1.0**
packaging: **zip**

To install the wls1211_dev.zip into the local maven repository, the mvn install:install-file goal is used.

```
$ mvn install:install-file -Dfile=wls1211_dev.zip -
DgroupId=com.oracle.weblogic -DartifactId=wls-dev -Dpackaging=zip -
Dversion=12.1.1.0

[INFO] Scanning for projects...
```

```
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building WebLogic Server Maven Plugin 12.1.1.0
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- maven-install-plugin:2.3.1:install-file (default-cli) @
wls-maven-plugin ---
[INFO] Installing /home/oracle/labs/Dev_labs_2012/WLS-
Maven/binaries/wls1211_dev.zip to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-
dev/12.1.1.0/wls-dev-12.1.1.0.zip
[INFO] Installing /tmp/mvninstall5298419836746462796.pom to
/home/oracle/.m2/repository/com/oracle/weblogic/wls-
dev/12.1.1.0/wls-dev-12.1.1.0.pom
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 26.869s
[INFO] Finished at: Mon Dec 19 19:49:05 PST 2011
[INFO] Final Memory: 2M/15M
[INFO] ------------------------------------------------------------
-----------
```

After the successful execution of this goal, the wls1211_dev.zip is now available as an artifact in the local maven repository with the following coordinates:

com.oracle.weblogic:wls-dev:zip:12.1.1.0

6. Install WebLogic Server 12*c*

The installation of wls1211_dev.zip can be performed using the maven wls:install goal.

The location of the distribution to install is specified in the pom.xml using the <artifactLocation> configuration element within the wls-maven-plugin definition.

```
<plugin>
  <groupId>com.oracle.weblogic</groupId>
  <artifactId>wls-maven-plugin</artifactId>
  <version>12.1.1.0</version>
  <configuration>
    <user>weblogic</user>
    <password>welcome1</password>
    <name>ConferencePlanner</name>
    <domainHome>
      ${project.build.directory}/domain
    </domainHome>
    <artifactLocation>
```

```
            com.oracle.weblogic:wls-dev:zip:12.1.1.0
          </artifactLocation>
        </configuration>
      </plugin>
```

The target directory for the installation will default to ${basedir}/Oracle/Software.

This can be overridden using the middleware home configuration property. For this lab, and for general usage, it's easier to use the default.

Note: the execution of the install goal may take several minutes since it needs to unzip, unpack then configure the resulting environment for use. This can be kept to a one-off cost if the WLS installation is not removed between maven executions. If the install goal detects the presence of a prior install, it will silently skip its execution.

To execute the installation, you can simply call the wls:install goal.

```
$ cd ~/labs/Dev_labs_2012/WLS-Maven/conference-planner/trunk

$ mvn wls:install

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:install (default-cli) @
ConferencePlanner ---
[INFO]
++================================================================++
[INFO] ++  wls-maven-plugin: install                              ++
[INFO]
++================================================================++
[INFO] Installing com.oracle.weblogic:wls-dev:zip:12.1.1.0 into:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/Oracle/Software
[INFO] Installing the product, this may take some time.
[INFO] Executing: [cmd:[/bin/bash, -c, chmod +x ./configure.sh;
./configure.sh]]
[INFO] Process being executed, waiting for completion.
[INFO] [exec] ************************************************
[INFO] [exec] WebLogic Server 12c (12.1.1.0) Zip Configuration
[INFO] [exec]
[INFO] [exec] MW_HOME:   /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/Oracle/Software
[INFO] [exec] JAVA_HOME: /labs/wls1211/jdk160_29
[INFO] [exec] ************************************************
[INFO] [exec]
```

```
[INFO] [exec] Please wait while 771 jars are unpacked ...
[INFO] [exec]
[INFO] [exec] Unpacking upgrade-launch.jar
770 to go
[INFO] [exec] Unpacking config-launch.jar
769 to go

...

[INFO] [exec] ...Unpacking done
[INFO] [exec]

...

[INFO] [exec]
[INFO] [exec] Your environment has been set.
[INFO] [exec] Configuring WLS...
[INFO] [exec]
INFO] ------------------------------------------------------------------
----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
----------
[INFO] Total time: 1:50.105s
[INFO] Finished at: Mon Dec 19 20:05:59 PST 2011
[INFO] Final Memory: 3M/15M
[INFO] ------------------------------------------------------------------
----------
```

The completed installation can be seen in the Oracle/Software subdirectory.

(fully qualified path: oracle/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-planner/trunk/Oracle/Software)

```
$ ls Oracle/Software

configure.cmd           configure.xml     modules
registry.template       utils             configure.sh
domain-registry.xml     README.txt        registry.xml
wlserver
```

7. Create a test domain

After the installation is complete, a domain can be created using the wls:create-domain goal. This goal will create a fully operational WLS domain that can be used to run an instance of WebLogic server.

The target directory for the domain will default to the ${basedir}/Oracle/Domains/mydomain directory.

This can be overridden using the domainHome configuration property.

Note: for this lab, the domain will be created in the ${project.build.directory}/domain directory. This allows the domain to be removed easily for future runs by executing the mvn clean goal, which removes the target directory.

The required username and password values for the new domain are specified as configuration values in the pom.xml file.

User: **weblogic**
Password: **welcome1**

Create the default domain using the wls:create-domain goal.

```
$ mvn wls:create-domain

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
------------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------
------------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:create-domain (default-cli) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: create-domain                     ++
[INFO]
++============================================================++
[INFO] Domain creation script:
readTemplate('/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/Oracle/Software/wlserver/common/templates/domains/wls.
jar')
cd('/Security/base_domain/User/weblogic')
set('Name', 'weblogic')
set('Password', '***')
writeDomain('/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain')
[INFO] ------------------------------------------------------------
------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
------------
[INFO] Total time: 29.717s
[INFO] Finished at: Mon Dec 19 20:22:06 PST 2011
[INFO] Final Memory: 18M/44M
[INFO] ------------------------------------------------------------
------------
```

The completed domain can be seen in the target/mydomain directory.

```
$ ls -l target/domain

total 80
drwxrwxr-x 2 oracle oracle 4096 Dec 19 20:22 autodeploy
drwxrwxr-x 5 oracle oracle 4096 Dec 19 20:22 bin
drwxrwxr-x 9 oracle oracle 4096 Dec 19 20:22 config
drwxrwxr-x 2 oracle oracle 4096 Dec 19 20:22 console-ext
-rw-rw-r-- 1 oracle oracle  462 Dec 19 20:22 fileRealm.properties
drwxrwxr-x 2 oracle oracle 4096 Dec 19 20:22 init-info
drwxrwxr-x 2 oracle oracle 4096 Dec 19 20:22 lib
drwxrwxr-x 2 oracle oracle 4096 Dec 19 20:22 security
drwxrwxr-x 3 oracle oracle 4096 Dec 19 20:22 servers
-rwxr-x--- 1 oracle oracle  296 Dec 19 20:22 startWebLogic.sh
```

8. Start the server

With the domain created, it can now be started using the wls:start-server goal.

Again, it's important to note that the configuration defaults and overrides are used consistently throughout the wls-maven-plugin so no additional configuration is necessary to identify the installation or domain to use.

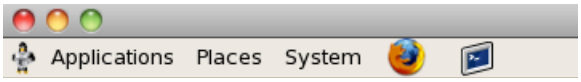
Start the server with the wls:start-server goal.

```
$ mvn wls:start-server

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:start-server (default-cli) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: start-server
++
[INFO]
++============================================================++
.[INFO] Starting server in domain:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain
[INFO] Check stdout file for details:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain/server-4206053735435021939.out
[INFO] Process being executed, waiting for completion.
.........
[INFO] Server started successful
[INFO] ------------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
```

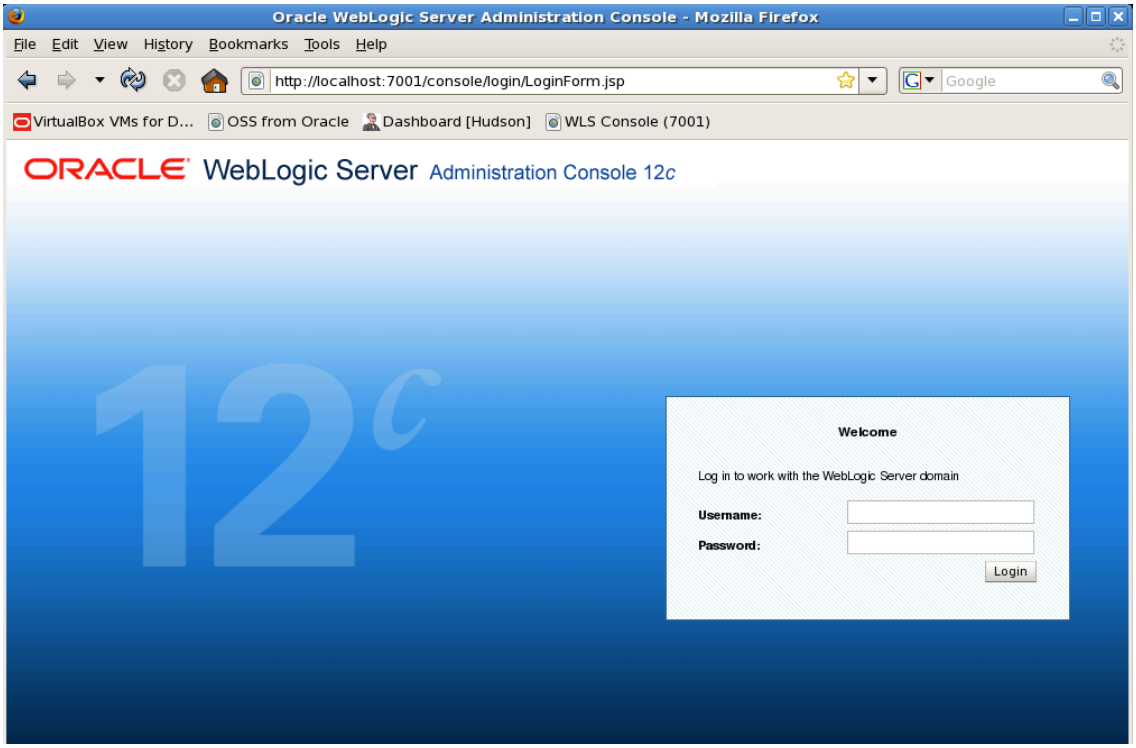
```
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.388s
[INFO] Finished at: Mon Dec 19 20:29:31 PST 2011
[INFO] Final Memory: 3M/15M
[INFO] ------------------------------------------------------------------------
```

You can check the server is now running by starting a browser using the menu bar icon.



And then accessing the WebLogic Server 12*c* Administration Console.

http://localhost:7001/console



9. Create a datasource using WLST

The wls-maven-plugin includes a set of goals that help you execute a number of operational and deployment operations on WebLogic Server.

A key operational feature of WebLogic Server is the WebLogic Server Scripting Tool (WLST) which enables common operations to be scripted using Jython and executed on a domain. The WLST environment has been made available to maven via the wls:wlst goal. This enables development and test environments to be setup and configured from within the maven lifecycle.

For this hands-on lab, you will use the wls:wlst goal to create a datasource required by the Conference Planner application.

The WLST script you will be executing is the src/main/wlst/create-derby-ds.py script.

Create the datasource using the wls:wlst goal, specifying the script to execute using the fileName system property.

```
$ mvn wls:wlst -DfileName=src/main/wlst/create-derby-ds.py

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:wlst (default-cli) @
ConferencePlanner ---
[INFO]
++================================================================
==++
[INFO] ++  wls-maven-plugin: wlst
++
[INFO]
++================================================================
==++

*** Creating DataSource ***

Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server 'AdminServer' that belongs to
domain 'domain'.

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

Location changed to edit tree. This is a writable tree with
DomainMBean as the root. To make changes you will need to start
an edit session via startEdit().

For more help, use help(edit)

Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Location changed to serverRuntime tree. This is a read-only tree
with ServerRuntimeMBean as the root.
```

```
For more help, use help(serverRuntime)


*** DataSource Details ***

  Name:            Derby Demo
  Driver Name:     Apache Derby Network Client JDBC Driver
  DataSource:      org.apache.derby.jdbc.ClientXADataSource
  Properties:      {serverName=localhost, portNumber=1527, user=app,
databaseName=demo;create=true}
  State:     Running
[INFO] ------------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
-----------
[INFO] Total time: 12.307s
[INFO] Finished at: Mon Dec 19 20:43:45 PST 2011
[INFO] Final Memory: 11M/28M
[INFO] ------------------------------------------------------------------
-----------
```

Note: for this simple example, the username and password of the target domain are specified in clear-text in the script that is executed. To avoid this level of exposure, the use of the WLST secure credential feature is highly recommended.

10. Deploy the application

With the server running and provisioned with the required data-source, the application can be deployed.

To deploy the application, the wls:deploy goal is used.

By convention the name of the application and the path to archive to deploy default to the calculated name of the application and archive that is produced by the packaging phase:

${project.build.directory}/${project.build.finalName}.${project.packaging}

This can be overridden using the name and source configuration element of the wls-maven-plugin configuration.

For this hands-on lab, we will simply use the defaults.

Deploy the application using the wls:deploy goal.

```
$ mvn wls:deploy

[INFO] Scanning for projects...
[INFO]
```

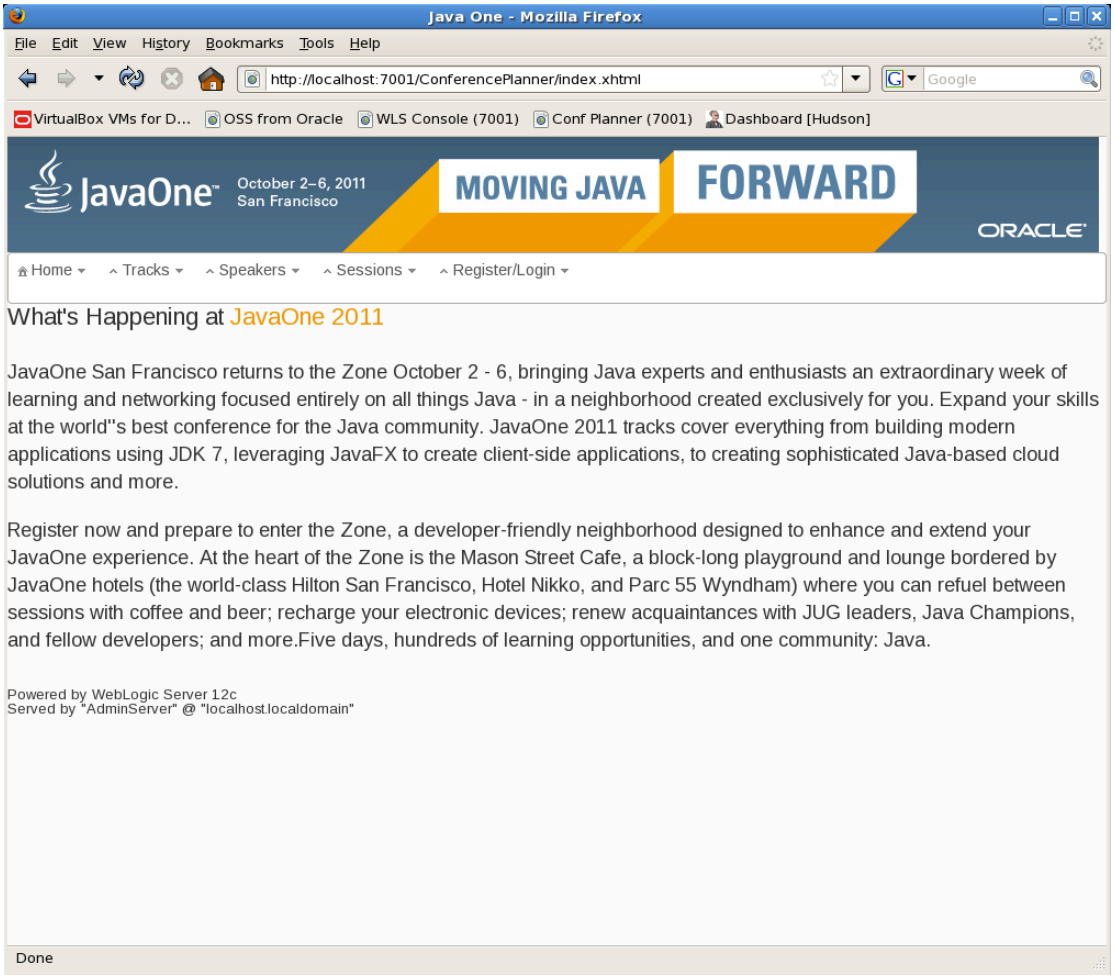
```
[INFO] ------------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:deploy (default-cli) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: deploy
++
[INFO]
++============================================================++
weblogic.Deployer invoked with options:  -noexit -user weblogic -
deploy -name ConferencePlanner -source
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/ConferencePlanner.war
<Dec 19, 2011 8:48:10 PM PST> <Info> <J2EE Deployment SPI> <BEA-
260121> <Initiating deploy operation for application,
ConferencePlanner [archive: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/ConferencePlanner.war], to
configured targets.>
Task 0 initiated: [Deployer:149026]deploy application
ConferencePlanner on AdminServer.
Task 0 completed: [Deployer:149026]deploy application
ConferencePlanner on AdminServer.
Target state: deploy completed on Server AdminServer

[INFO] ------------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------
-----------
[INFO] Total time: 7.299s
[INFO] Finished at: Mon Dec 19 20:48:21 PST 2011
[INFO] Final Memory: 5M/15M
[INFO] ------------------------------------------------------------------
-----------
```

11. Test the application using a browser

With the application successfully deployed, it can be tested immediately using the browser.

http://localhost:7001/ConferencePlanner



Using the menu bar, you can navigate your way around the application, viewing the different tracks and sessions that were presented at JavaOne 2011.

For the keen, you can use the Register/Login action to create a new user and create yourself a session schedule. Just don't turn up, the conference is long over ... ☺

12. Shutdown and clean up

The wls-maven-plugin goals can be specified as part of a multiple goal sequence, so they are invoked in turn from the one maven execution.

To clean up your test environment, you can use the wls:stop-server goal to stop the running server, and then the maven clean goal to remove the target directory. Since the domain is resident in the target directory, it will be removed along with the build artifacts of the application.

Note: if you want your domain to **not** be cleaned up with the application build artifacts when the maven clean goal is executed, create the domain outside of the target directory.

Stop the server with wls:stop-server and clean the project.

```
$ mvn wls:stop-server clean

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:stop-server (default-cli) @
ConferencePlanner ---
[INFO]
++================================================================++
[INFO] ++  wls-maven-plugin: stop-server                          ++
[INFO]
++================================================================++
[INFO] Stop server in domain: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/domain
[INFO] Process being executed, waiting for completion.
[INFO] [exec] Stopping Weblogic Server...
[INFO] [exec]
[INFO] [exec] Initializing WebLogic Scripting Tool (WLST) ...
[INFO] [exec]
[INFO] [exec] Welcome to WebLogic Server Administration Scripting
Shell
[INFO] [exec]
[INFO] [exec] Type help() for help on available commands
[INFO] [exec]
[INFO] [exec] Connecting to t3://localhost:7001 with userid weblogic
...
[INFO] [exec] Successfully connected to Admin Server 'AdminServer'
that belongs to domain 'domain'.
[INFO] [exec]
[INFO] [exec] Warning: An insecure protocol was used to connect to
the
[INFO] [exec] server. To ensure on-the-wire security, the SSL port
or
[INFO] [exec] Admin port should be used instead.
[INFO] [exec]
[INFO] [exec] Shutting down the server AdminServer with force=false
while connected to AdminServer ...
[INFO] [exec] Disconnected from weblogic server: AdminServer
[INFO] [exec]
[INFO] [exec]
[INFO] [exec] Exiting WebLogic Scripting Tool.
[INFO] [exec]
[INFO] [exec] Done
[INFO] [exec] Stopping Derby Server...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @
ConferencePlanner ---
```

```
[INFO] Deleting /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 6.547s
[INFO] Finished at: Mon Dec 19 21:17:56 PST 2011
[INFO] Final Memory: 3M/15M
[INFO] ------------------------------------------------------------
-----------
```

If you now try and start the server, you should see an error since the domain has been removed.

```
$ mvn wls:start-server

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------
-----------
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:start-server (default-cli) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: start-server
++
[INFO]
++============================================================++
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 0.645s
[INFO] Finished at: Mon Dec 19 21:19:45 PST 2011
[INFO] Final Memory: 3M/15M
[INFO] ------------------------------------------------------------
-----------
[ERROR] Failed to execute goal com.oracle.weblogic:wls-maven-
plugin:12.1.1.0:start-server (default-cli) on project
ConferencePlanner: Error starting server: No such file or directory
-> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with
the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug
logging.
[ERROR]
```

```
[ERROR] For more information about the errors and possible
solutions, please read the following articles:
[ERROR] [Help 1]
http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureExceptio
n
```

Fear not! The domain can be easily created again using the wls:create-domain goal.

13. Execute a full end-to-end lifecycle

Using standard maven features, the wls-maven-plugin goals can be mapped to automatically execute when specific maven phases are called.

A number of the goals are pre-defined with default phase mappings:

pre-integration-test:	install, create-domain, start-server
post-integration-test:	undeploy, stop-server

This means that these goals will automatically execute in the defined phase when they are specified as execution points in the pom.xml file.

The wlst goal in contrast is not mapped by default to any specific maven phase since it can be used across a number of phases.

```
<executions>
  <execution>
    <id>WLS_SETUP</id>
    <goals>
       <goal>install</goal>
      <goal>create-domain</goal>
      <goal>start-server</goal>
    </goals>
  </execution>

  <execution>
    <id>WLS_SETUP_RESOURCES</id>
    <!--
    WLST doesn't have a phase binding since it
    can be used pre/post integration-test, so need
    to specifically call it out
    -->
    <phase>pre-integration-test</phase>
    <goals>
      <goal>wlst</goal>
    </goals>
    <configuration>
      <fileName>
        ${basedir}/src/main/wlst/create-derby-ds.py</fileName>
```

```
        </configuration>
      </execution>
      ...
    </executions>
```

For this hands-on lab, a sequence of wls-maven-plugin execution points have been configured in a maven profile called WLS. When maven is executed, a profile to use can be specified, whereupon maven will use the configuration elements specified in that specific profile.

Using the WLS profile, you can execute a complete end-to-end lifecycle, which will build, package, setup WLS, deploy the application, execute a simple integration test and then clean everything up.

Note: the WLS profile is not a default option provided by the wls-maven-plugin, it is just an example of using a standard maven profile, to which we've ascribed the name WLS ... for obvious reasons I guess.

To execute the WLS profile, execute maven with a "–P WLS" option to specify the use of the WLS profile. A complete build, test lifecycle will run, where the testing of the application occurs on the freshly created and provisioned WLS instance.

```
$ mvn –P WLS

[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------
-----------
[INFO] Building ConferencePlanner 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------
-----------
[INFO]
[INFO] --- maven-dependency-plugin:2.1:copy (default) @
ConferencePlanner ---
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:resources (default-
resources) @ ConferencePlanner ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @
ConferencePlanner ---
[INFO] Compiling 18 source files to
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:testResources (default-
testResources) @ ConferencePlanner ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/src/test/resources
[INFO]
```

```
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-
testCompile) @ ConferencePlanner ---
[INFO] Compiling 3 source files to
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.5:test (default-test) @
ConferencePlanner ---
[INFO] Surefire report directory:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
There are no tests to run.

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-war-plugin:2.1.1:war (default-war) @
ConferencePlanner ---
[INFO] Packaging webapp
[INFO] Assembling webapp [ConferencePlanner] in
[/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/ConferencePlanner]
[INFO] Processing war project
[INFO] Copying webapp resources
[/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/src/main/webapp]
[INFO] Webapp assembled in [79 msecs]
[INFO] Building war: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/ConferencePlanner.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:install (WLS_SETUP) @
ConferencePlanner ---
[INFO]
++======================================================================++
[INFO] ++  wls-maven-plugin: install                                    ++
[INFO]
++======================================================================++
[INFO] Installing com.oracle.weblogic:wls-dev:zip:12.1.1.0 into:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/Oracle/Software
[INFO] Skipping install, previous installation found at
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/Oracle/Software
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:create-domain (WLS_SETUP) @
ConferencePlanner ---
[INFO]
++======================================================================++
[INFO] ++  wls-maven-plugin: create-domain                              ++
[INFO]
++======================================================================++
[INFO] Domain creation script:
```

```
readTemplate('/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/Oracle/Software/wlserver/common/templates/domains/wls.
jar')
cd('/Security/base_domain/User/weblogic')
set('Name', 'weblogic')
set('Password', '***')
writeDomain('/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain')
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:start-server (WLS_SETUP) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: start-server                      ++
[INFO]
++============================================================++
[INFO] Starting server in domain:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain
[INFO] Check stdout file for details:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/domain/server-3437835154690502782.out
[INFO] Process being executed, waiting for completion.
.........
[INFO] Server started successful
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:wlst (WLS_SETUP_RESOURCES) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: wlst                              ++
[INFO]
++============================================================++

*** Creating DataSource ***

Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server 'AdminServer' that belongs to
domain 'domain'.

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

Location changed to edit tree. This is a writable tree with
DomainMBean as the root. To make changes you will need to start
an edit session via startEdit().

For more help, use help(edit)

Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
Location changed to serverRuntime tree. This is a read-only tree
with ServerRuntimeMBean as the root.
```

```
For more help, use help(serverRuntime)


*** DataSource Details ***

  Name:            Derby Demo
  Driver Name:     Apache Derby Network Client JDBC Driver
  DataSource:      org.apache.derby.jdbc.ClientXADataSource
  Properties:      {serverName=localhost, portNumber=1527, user=app,
databaseName=demo;create=true}
  State:     Running
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:deploy (WLS_DEPLOY) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: deploy                             ++
[INFO]
++============================================================++
weblogic.Deployer invoked with options:  -noexit -user weblogic -
deploy -name ConferencePlanner -source
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/ConferencePlanner.war
<Dec 19, 2011 9:35:53 PM PST> <Info> <J2EE Deployment SPI> <BEA-
260121> <Initiating deploy operation for application,
ConferencePlanner [archive: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/ConferencePlanner.war], to
configured targets.>
Task 0 initiated: [Deployer:149026]deploy application
ConferencePlanner on AdminServer.
Task 0 completed: [Deployer:149026]deploy application
ConferencePlanner on AdminServer.
Target state: deploy completed on Server AdminServer

[INFO]
[INFO] --- maven-failsafe-plugin:2.11:integration-test (default) @
ConferencePlanner ---
[INFO] Failsafe report directory:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/failsafe-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running sab.test.conference.VerifyWebLogicDescriptorIT
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.02
sec
Running sab.test.conference.SimulateErrorIT
Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.001 sec
Running sab.test.conference.VerifyGlassFishDescriptorIT
Tests run: 2, Failures: 0, Errors: 0, Skipped: 2, Time elapsed:
0.001 sec

Results :

Tests run: 5, Failures: 0, Errors: 0, Skipped: 3
```

```
[WARNING] File encoding has not been set, using platform encoding
UTF-8, i.e. build is platform dependent!
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:wlst (WLS_TEARDOWN_RESOURCES) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: wlst                               ++
[INFO]
++============================================================++
Connecting to t3://localhost:7001 with userid weblogic ...
Successfully connected to Admin Server 'AdminServer' that belongs to
domain 'domain'.

Warning: An insecure protocol was used to connect to the
server. To ensure on-the-wire security, the SSL port or
Admin port should be used instead.

Location changed to edit tree. This is a writable tree with
DomainMBean as the root. To make changes you will need to start
an edit session via startEdit().

For more help, use help(edit)

Starting an edit session ...
Started edit session, please be sure to save and activate your
changes once you are done.
None
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released
once the activation is completed.
Activation completed
[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:undeploy (WLS_TEARDOWN) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: undeploy                           ++
[INFO]
++============================================================++
weblogic.Deployer invoked with options:  -noexit -user weblogic -
undeploy -name ConferencePlanner
<Dec 19, 2011 9:36:15 PM PST> <Info> <J2EE Deployment SPI> <BEA-
260121> <Initiating undeploy operation for application,
ConferencePlanner [archive: null], to configured targets.>
Task 1 initiated: [Deployer:149026]remove application
ConferencePlanner on AdminServer.
Task 1 completed: [Deployer:149026]remove application
ConferencePlanner on AdminServer.
Target state: undeploy completed on Server AdminServer

[INFO]
[INFO] --- wls-maven-plugin:12.1.1.0:stop-server (WLS_TEARDOWN) @
ConferencePlanner ---
[INFO]
++============================================================++
[INFO] ++  wls-maven-plugin: stop-server                        ++
[INFO]
++============================================================++
```

```
[INFO] Stop server in domain: /home/oracle/labs/Dev_labs_2012/WLS-
Maven/conference-planner/trunk/target/domain
[INFO] Process being executed, waiting for completion.
[INFO] [exec] Stopping Weblogic Server...
[INFO] [exec]
[INFO] [exec] Initializing WebLogic Scripting Tool (WLST) ...
[INFO] [exec]
[INFO] [exec] Welcome to WebLogic Server Administration Scripting
Shell
[INFO] [exec]
[INFO] [exec] Type help() for help on available commands
[INFO] [exec]
[INFO] [exec] Connecting to t3://localhost:7001 with userid weblogic
...
[INFO] [exec] Successfully connected to Admin Server 'AdminServer'
that belongs to domain 'domain'.
[INFO] [exec]
[INFO] [exec] Warning: An insecure protocol was used to connect to
the
[INFO] [exec] server. To ensure on-the-wire security, the SSL port
or
[INFO] [exec] Admin port should be used instead.
[INFO] [exec]
[INFO] [exec] Shutting down the server AdminServer with force=false
while connected to AdminServer ...
<Dec 19, 2011 9:36:21 PM PST> <Warning> <JNDI> <BEA-050001>
<WLContext.close() was called in a different thread than the one in
which it was created.>
<Dec 19, 2011 9:36:21 PM PST> <Warning> <JNDI> <BEA-050001>
<WLContext.close() was called in a different thread than the one in
which it was created.>
WLST lost connection to the WebLogic Server that you were
connected to, this may happen if the server was shutdown or
partitioned. You will have to re-connect to the server once the
server is available.
Disconnected from weblogic server: AdminServer
[INFO] [exec] WLST lost connection to the WebLogic Server that you
were
[INFO] [exec] connected to, this may happen if the server was
shutdown or
[INFO] [exec] partitioned. You will have to re-connect to the server
once the
[INFO] [exec] server is available.
[INFO] [exec] Disconnected from weblogic server: AdminServer
WLST lost connection to the WebLogic Server that you were
connected to, this may happen if the server was shutdown or
partitioned. You will have to re-connect to the server once the
server is available.
Disconnected from weblogic server: AdminServer
[INFO] [exec] Disconnected from weblogic server:
[INFO] [exec]
[INFO] [exec]
[INFO] [exec] Exiting WebLogic Scripting Tool.
[INFO] [exec]
[INFO] [exec] Done
[INFO] [exec] Stopping Derby Server...
[INFO]
[INFO] --- maven-failsafe-plugin:2.11:verify (default) @
ConferencePlanner ---
```

```
[INFO] Failsafe report directory:
/home/oracle/labs/Dev_labs_2012/WLS-Maven/conference-
planner/trunk/target/failsafe-reports
[WARNING] File encoding has not been set, using platform encoding
UTF-8, i.e. build is platform dependent!
[INFO] ------------------------------------------------------------
-----------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
-----------
[INFO] Total time: 1:23.092s
[INFO] Finished at: Mon Dec 19 21:36:24 PST 2011
[INFO] Final Memory: 41M/103M
[INFO] ------------------------------------------------------------
-----------
```

This example of using the WLS profile demonstrates the automation of a complete dev, deploy and test lifecycle using WebLogic Server 12*c* as the server under test.

Summary

The wls-maven-plugin provides extensive support for incorporating WebLogic Server into full end-to-end development/testing lifecycles. Using sensible, consistent defaults across the goals, the amount of configuration required is reduced to one or two mandatory values.

wls:install	Installs WebLogic Server
wls:create-domain	Creates a domain
wls:start-server	Starts the server in the domain under test
wls:deploy	Deploys the application
wls:stop-server	Stops the server in the domain under test

The wls-maven-plugin contains more goals than has been demonstrated in this handson-lab, including the ability to execute the appc utility on the application to pre-compile it prior to deployment.

Part 3: Setting up a Continuous Integration Environment with Hudson, Maven and WebLogic Server 12*c*

Overview

In part 2 of this hands-on lab, you explored the use of the wls-maven-plugin and observed how it enables the incorporation of WebLogic Server into an end-to-end dev to test lifecycle.

In part 3, you will leverage the environment from part 2 but now inject the Hudson Continuous Integration Server into the architecture to create a fully automated, continuous testing environment.

Through the use of Hudson, you will be able to continuously monitor the progress and quality of the development activity of the application by creating a build job that uses our Conference Planner maven project.

As you have just seen, the Conference Planner maven project is configured to perform a completely automated development, deploy and test lifecycle. By configuring the Hudson job to watch the subversion repository for changes, updating itself with the change set, then executing the WLS profile defined on the maven project, a rapid feedback cycle will be created that shows the test results when changes are checked in.

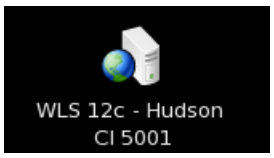
In this lab, you will deploy the Hudson application to a WebLogic Server domain; using the Hudson dashboard, create a job to watch, build and test the Conference Planner application; manually execute a build to observe the execution and verify it works; and finally introduce a small change to the application, check it into the subversion repository and observe the impact of the change through the Hudson dashboard.

1. Start the WebLogic Server 12*c* – Hudson CI domain

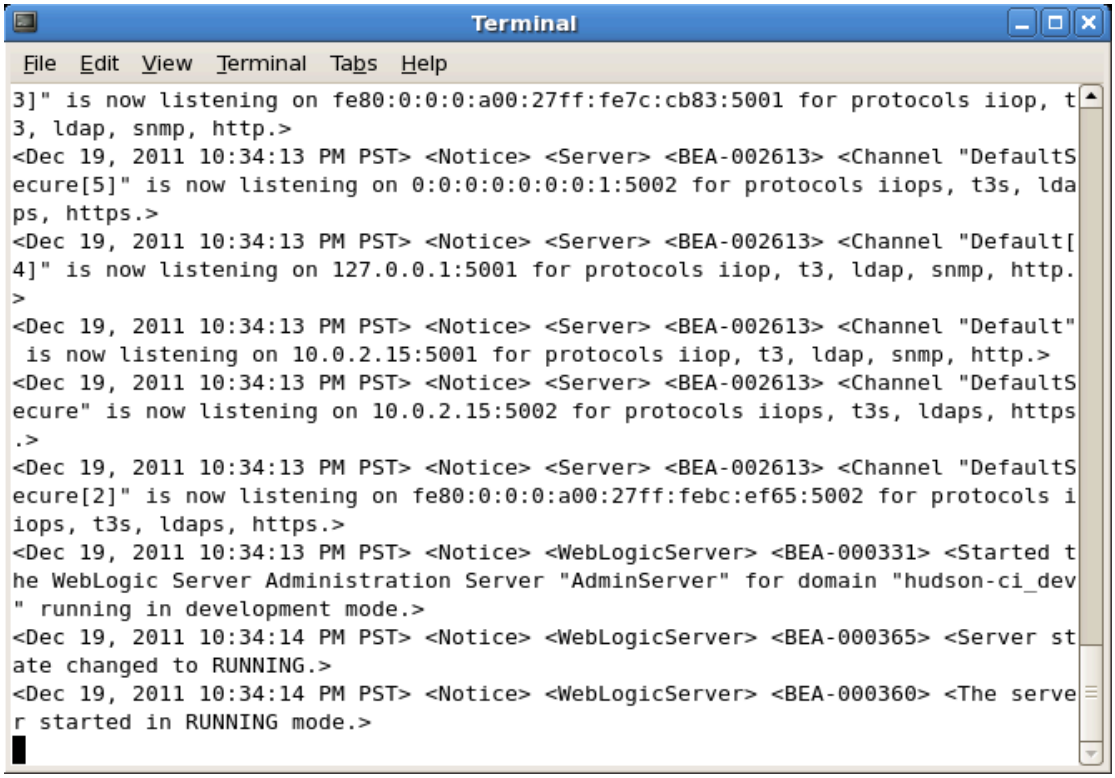
An empty domain has been created for you to use in this section of the hands-on lab.

Name:	**hudson-ci_dev**
Port:	**5001**
Username:	**weblogic**
Password:	**welcome1**

Locate the WLS 12*c* – Hudson CI icon on the desktop and click it to start the server.



This will start the server in the hudson-ci_dev domain.

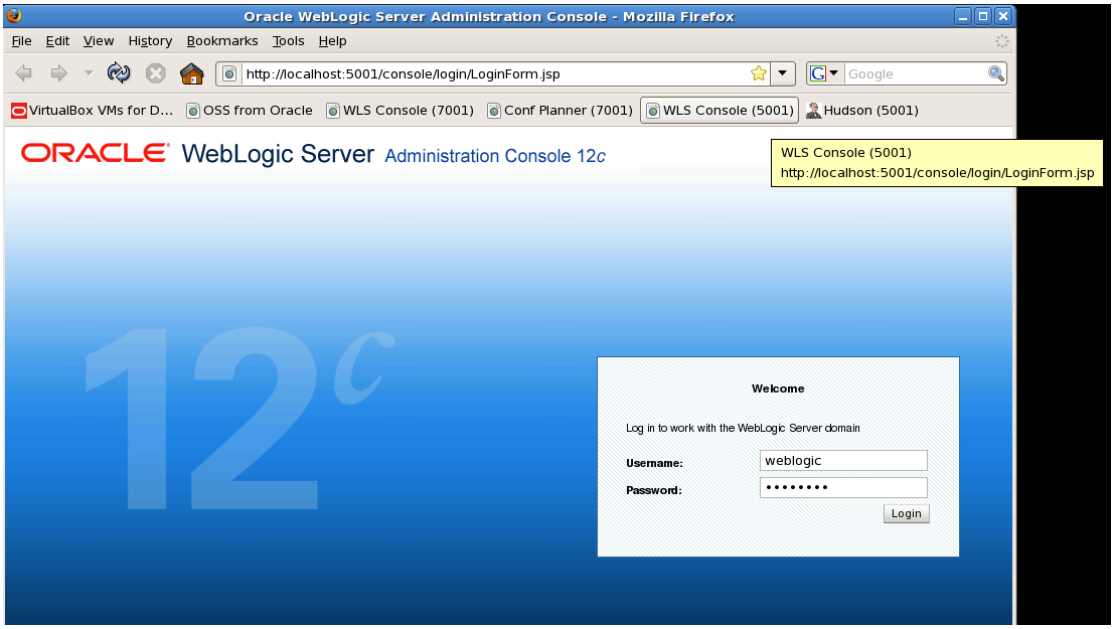


2. Deploy the Hudson 2.2.0 application

The Hudson Continuous Integration server is distributed as a Web Application Archive (war) and can be deployed to Java EE compatible servers.

To deploy Hudson to WebLogic Server, open a browser, access the WebLogic Server Administration Console and login using the credentials shown above.

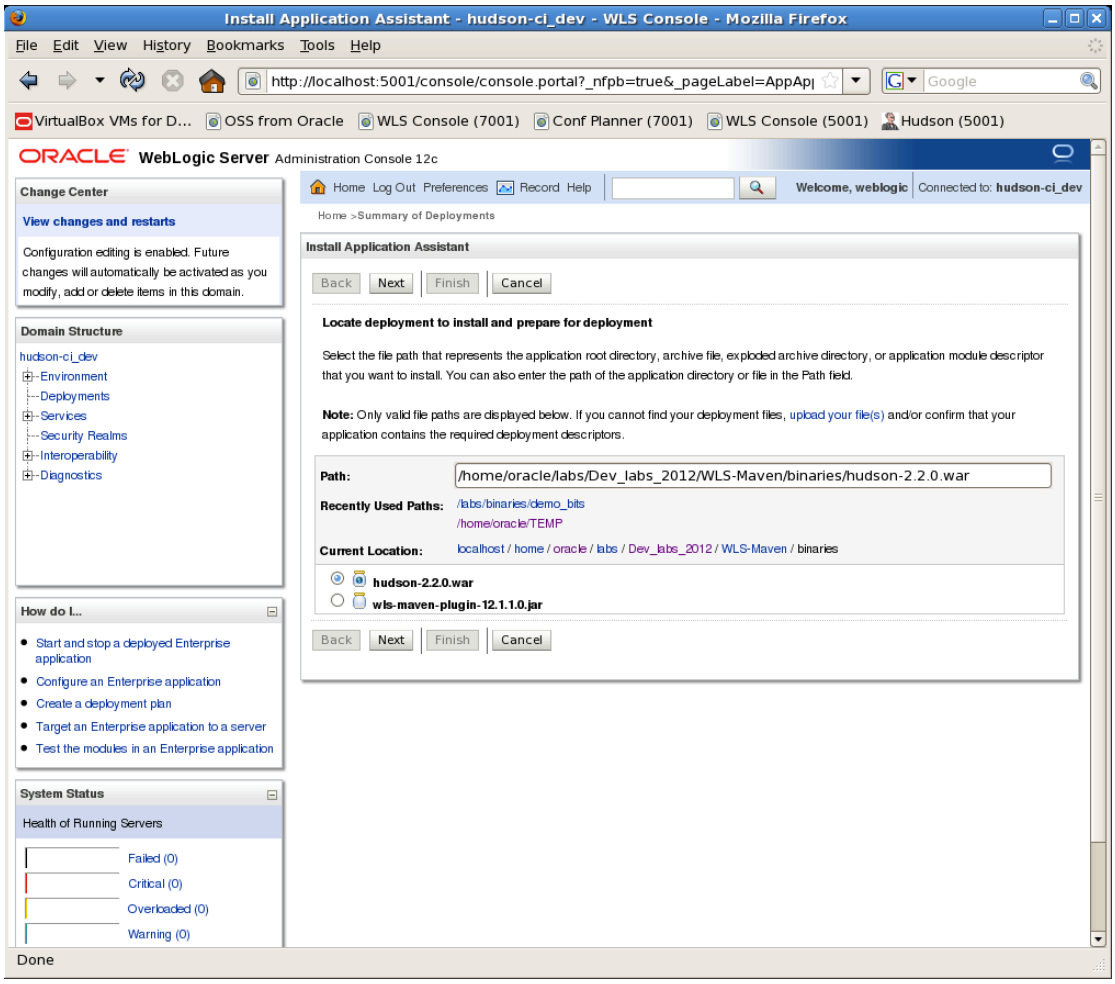
http://localhost:5001/console



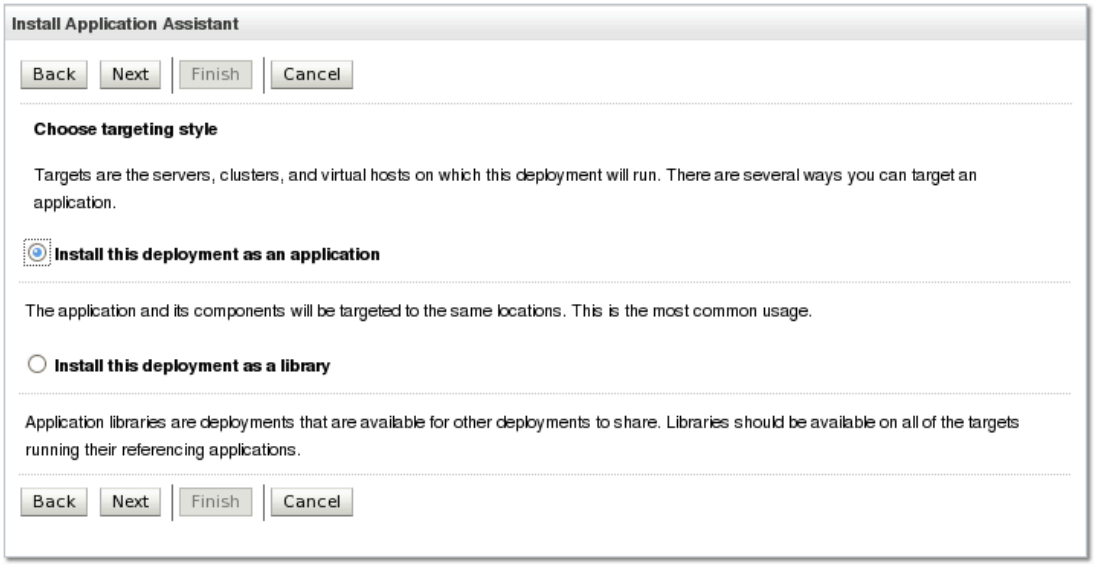
Select **Deployments** > **Install** to initiate the deployment wizard.

Using the path browser, browse to the **/home/oracle/labs/Dev_lab_2012/WLS-Maven/binaries** directory,

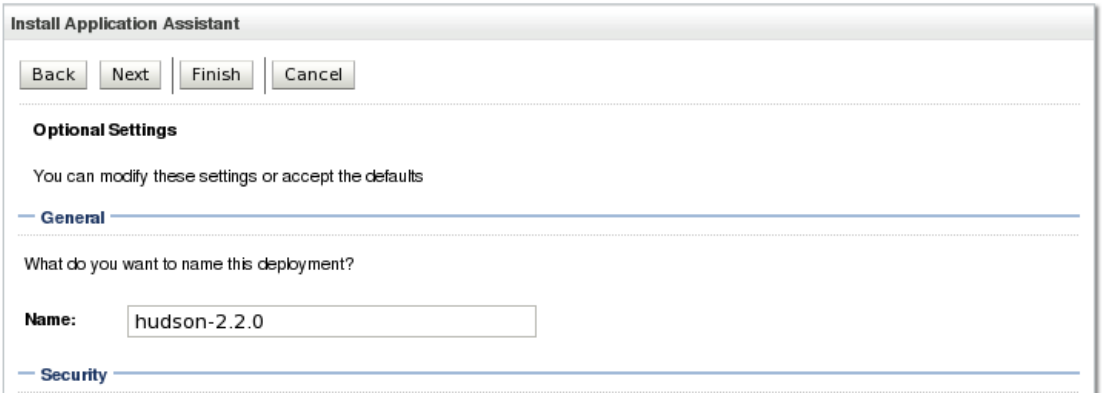
Select the **hudson-2.2.0.war** file, click **Next**.



Select **Install this deployment as an application** and click **Next**

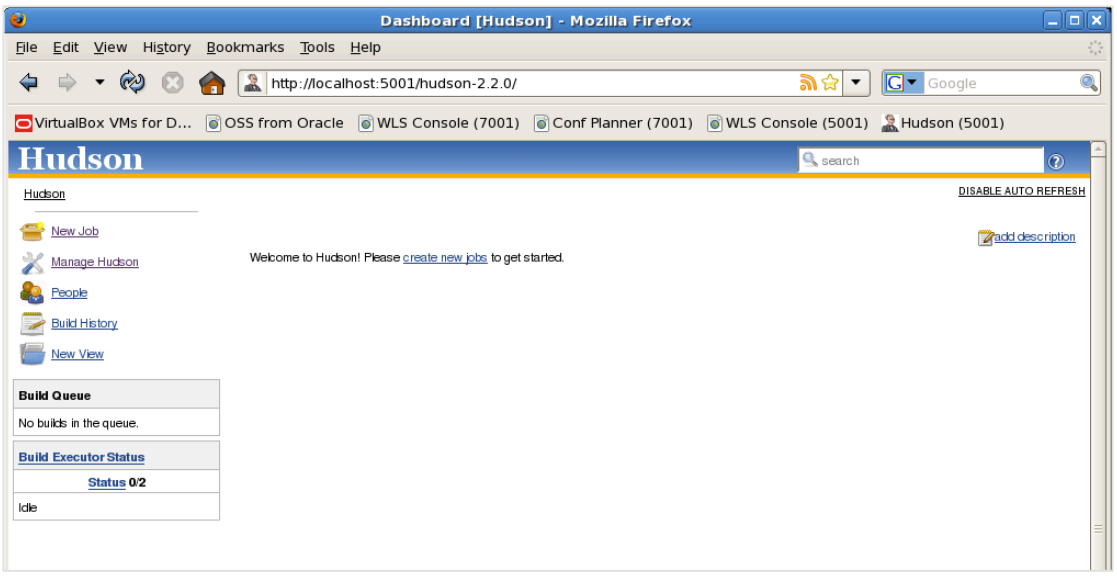


Select **Finish** to complete the deployment operation.



Once the deployment has been completed, now test that the deployment was successful by accessing the Hudson dashboard from the browser.

http://localhost:5001/hudson-2.2.0



My name is Hudson, I'm here to serve!

3. Configure the conference planner job

Let's give Hudson some work to do.

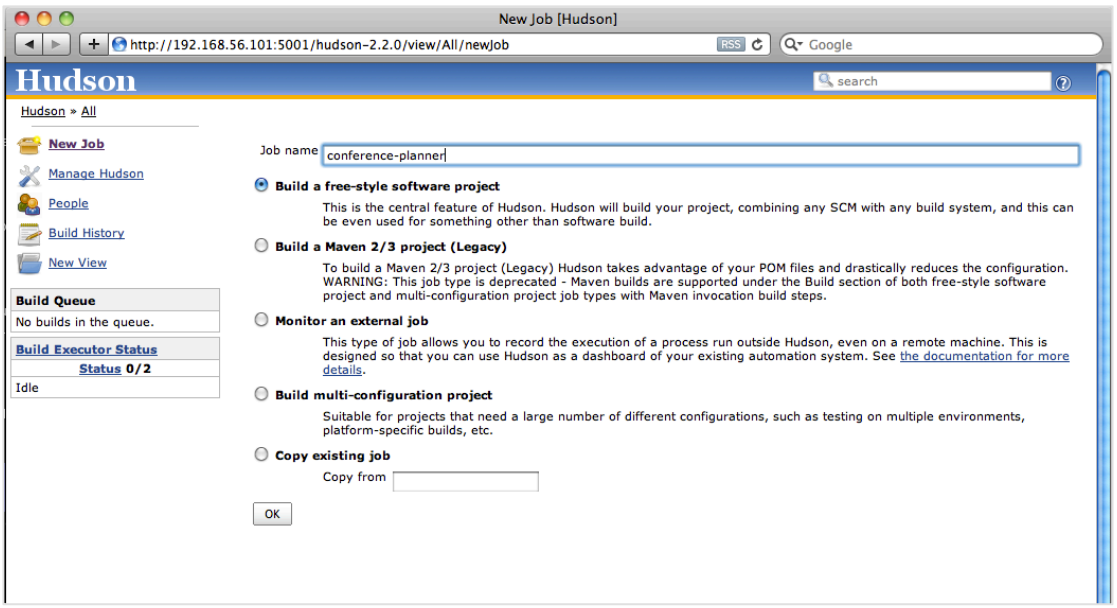
The goal of this step is to create a new Hudson job that builds using maven 3 and which is based on the same conference-planner project you checked out from the local subversion repository in part 2.

You will want to configure the job so that it executes an end-to-end lifecycle that builds, packages, deploys and tests the application and you'll want Hudson to display the test results.

Click the **create new jobs** link to create a new job.

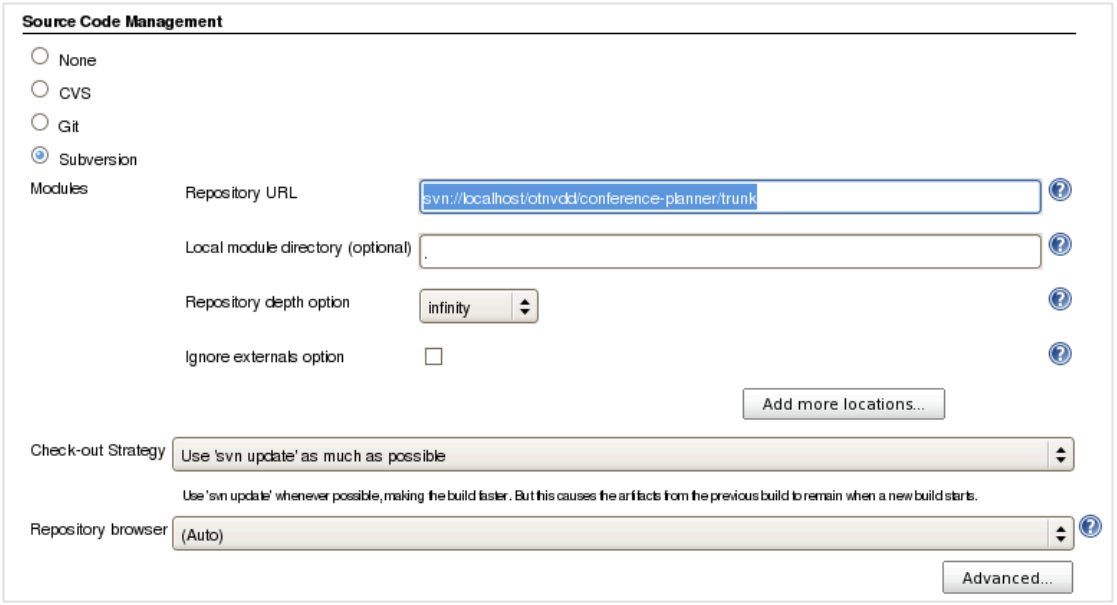
Specify the job name as **conference-planner** and select the **Build a free-style software project** option.

Note: it's very important you use the **exact** name conference-planner in order for the subversion notifications to work.



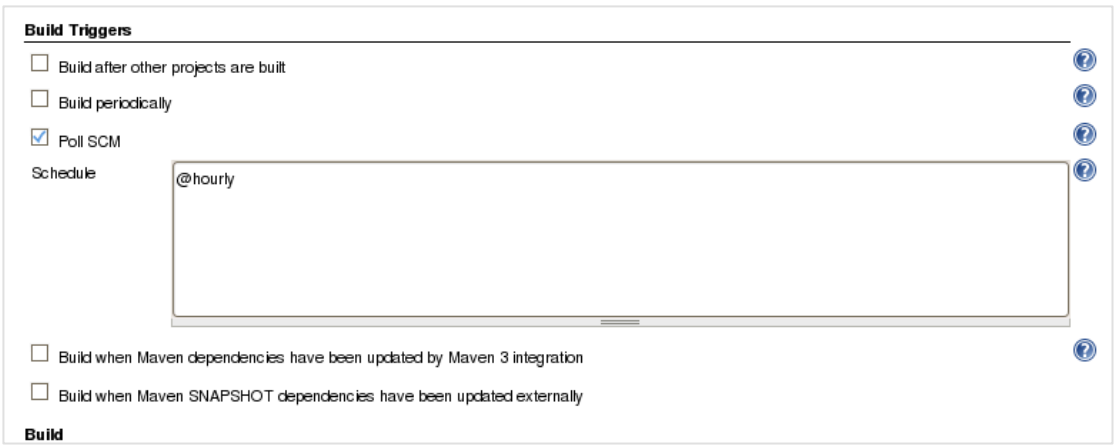
Configure the **Source Code Management** option by selecting the **Subversion** option and specifying the repository URL as:

svn://localhost/otnvdd/conference-planner/trunk

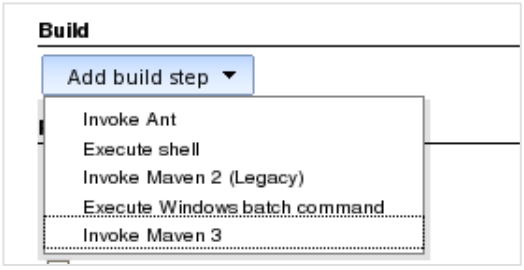


Configure the **Build Trigger** by selecting the **Poll** option and specifying a value of **@hourly**.

Note: we don't really want to poll frequently, but this option needs to be enabled to allow the subversion notifications to work. The integration works by subversion issuing a HTTP request to the Hudson job to ask it to poll and see if a change needs to be tested.

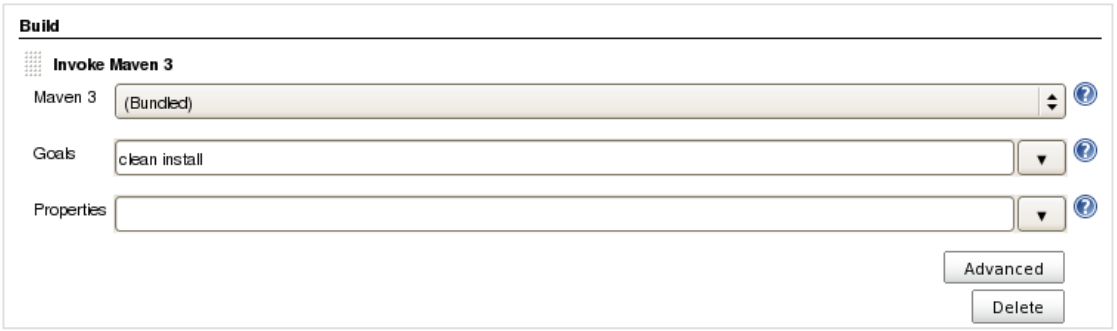


Add a **Build Step** by clicking the **Add Build Step** button and selecting the **Invoke Maven 3** option.



The default settings can remain.

For the same **Build Step**, click the **Advanced** button to specify some additional configuration settings for this maven build operation.

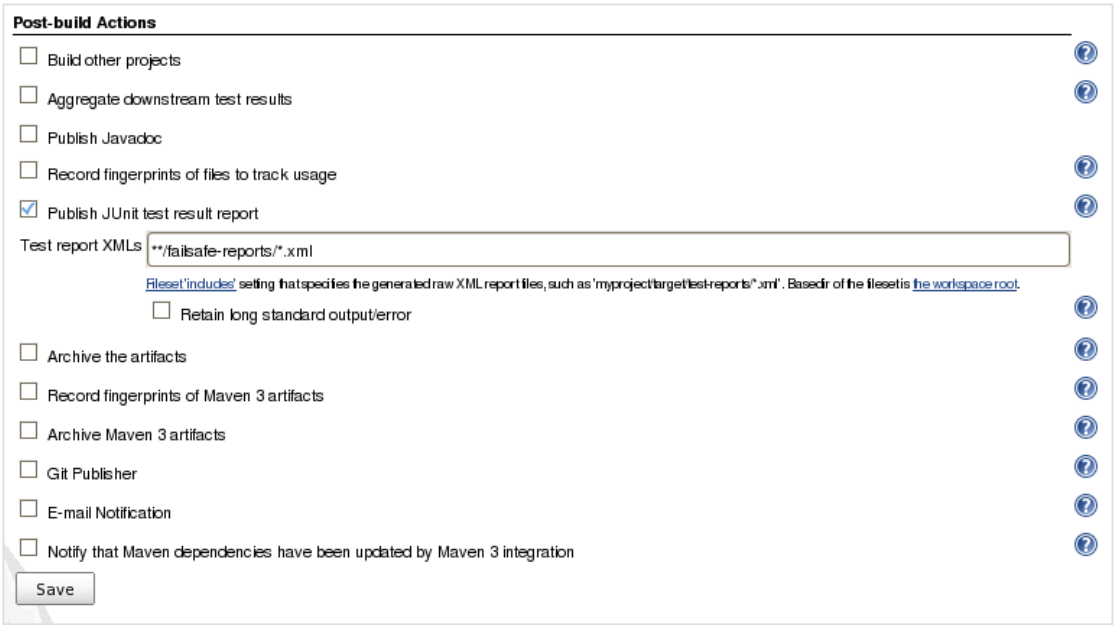


In the advanced configuration, locate the properties shown below and enter the specified values:

Profiles: **WLS**
JVM Options: **-Xmx512m -XX:MaxPermSize=256m**

As the final configuration step for this job, locate the **Post-build Actions** section, check the **Publish JUnit test result report** and enter the following value in the **Test report XMLs** field:

Test report XMLs: **/failsafe-reports/*.xml

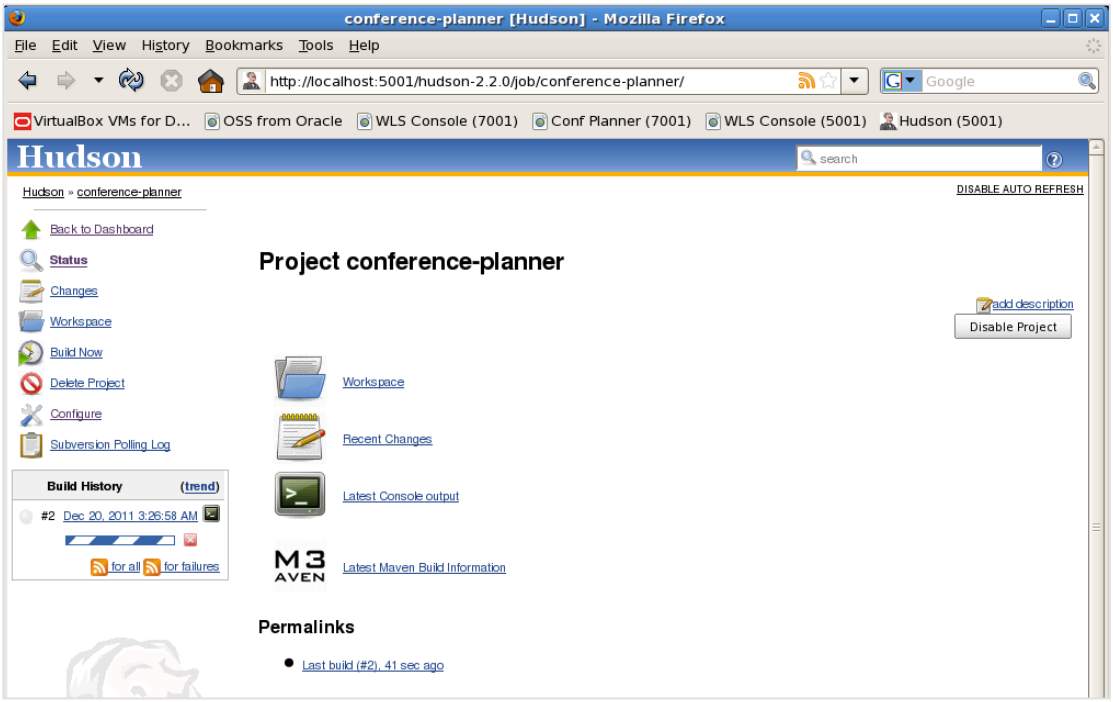


Click the **Save** button to complete the job configuration.

4. Execute a build, view console output and test results

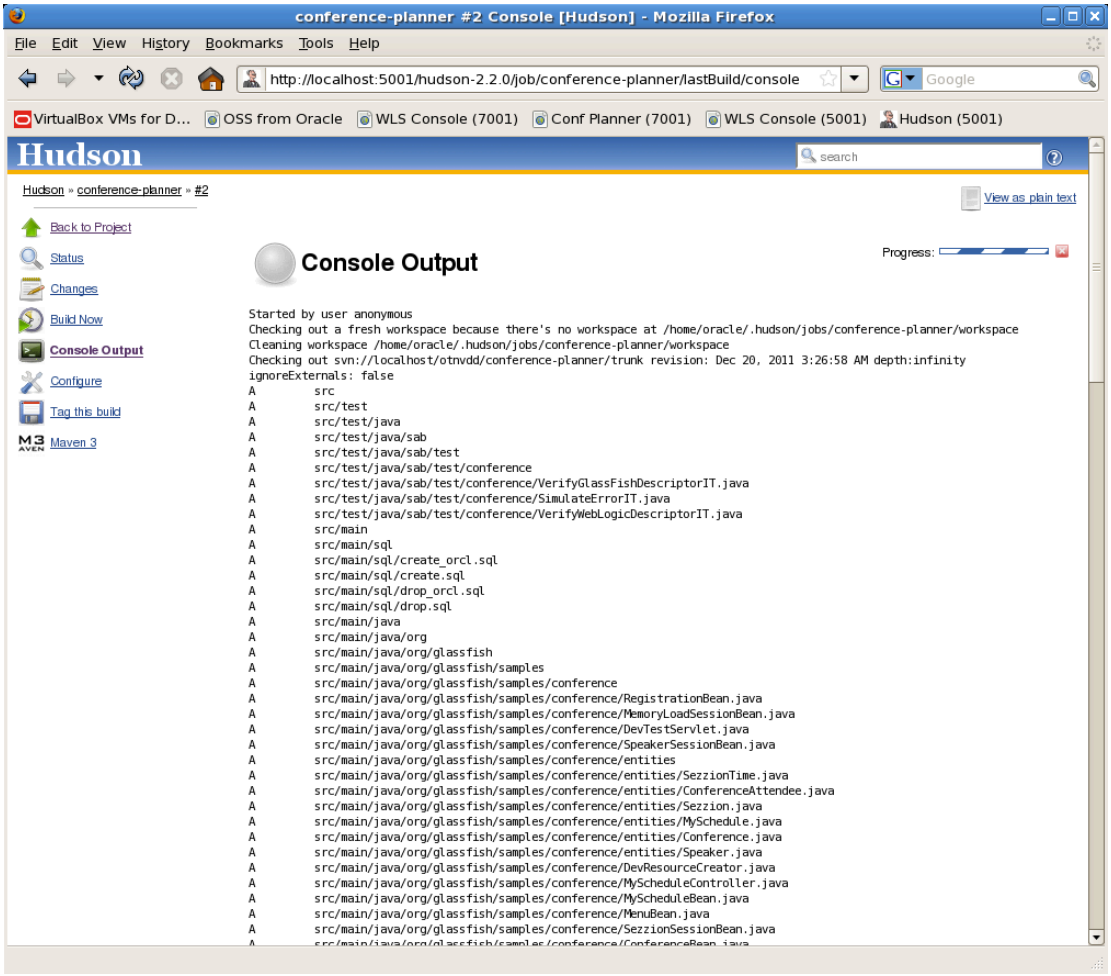
To validate the correctness of the job, you can now manually trigger a build to be done and observe the results.

From the Configuration Planner job page, click the **Build Now** icon. A build will start as shown in the **Build History** portlet.



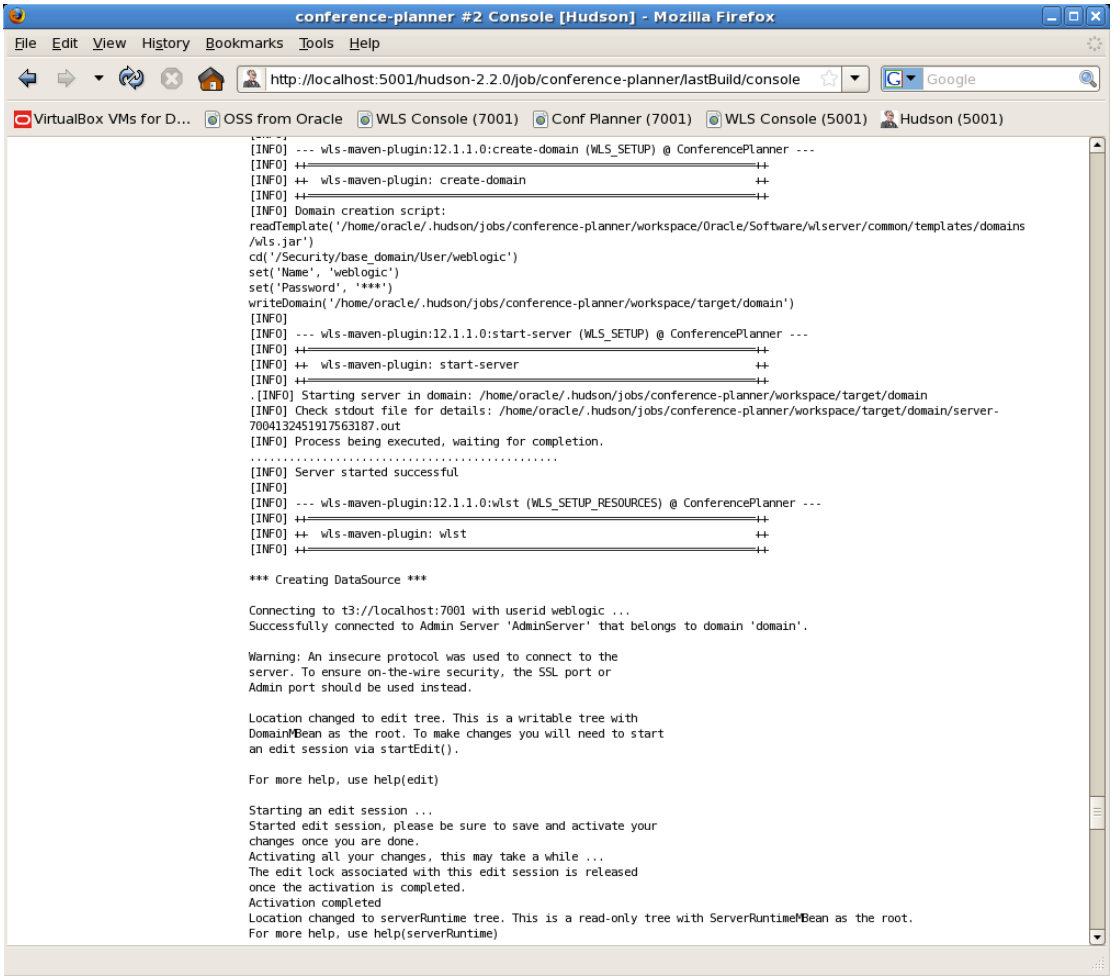
To view the progress of the build, click the **Latest Console output** link to see the log messages and other standard output from the job as it executes.

You should see at the top of the console output messages indicating that the job is checking out a fresh workspace and then checks out the project from the specified repository location.



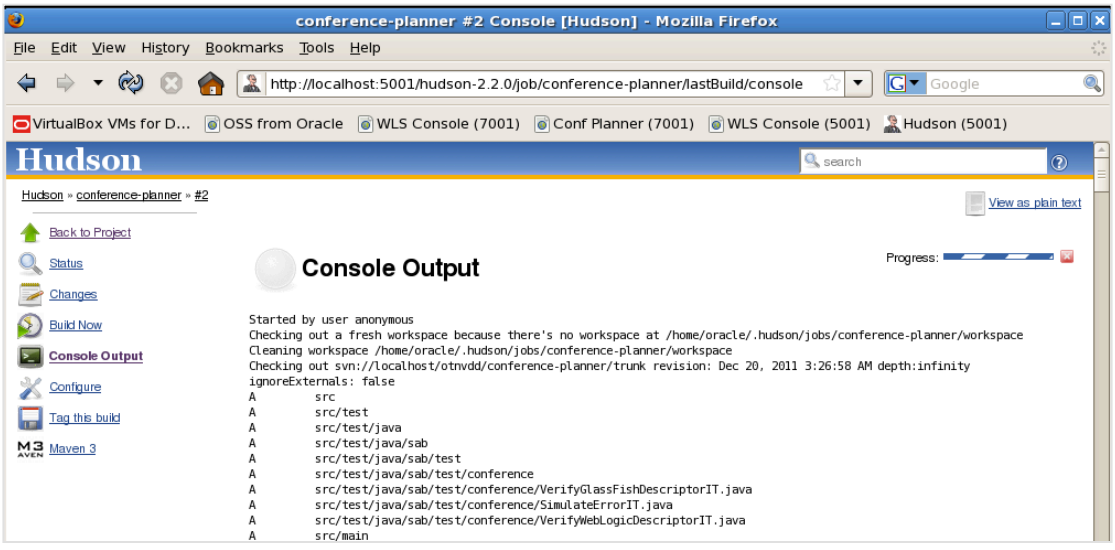
The maven project will execute using the same WLS profile that you observed in part 2 when you executed it from the command line.

This will install wls1211_dev.zip using the same zip file you installed into the maven repository in part 2; create a domain and start it; create a datasource against the local derby database; deploy the application and execute a small set of tests; undeploy the application and remove the datasource; stop the running server.



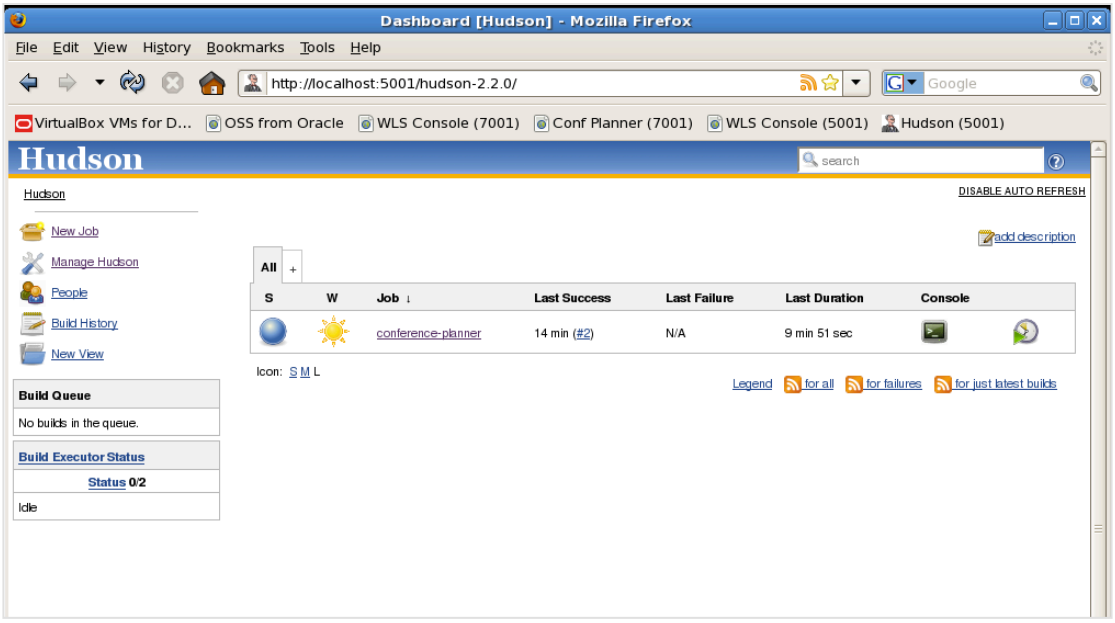
Once the job completes, the final status will be reflected back on the Hudson dashboard.

To access the dashboard, click the **Hudson** link at the top left of the page.

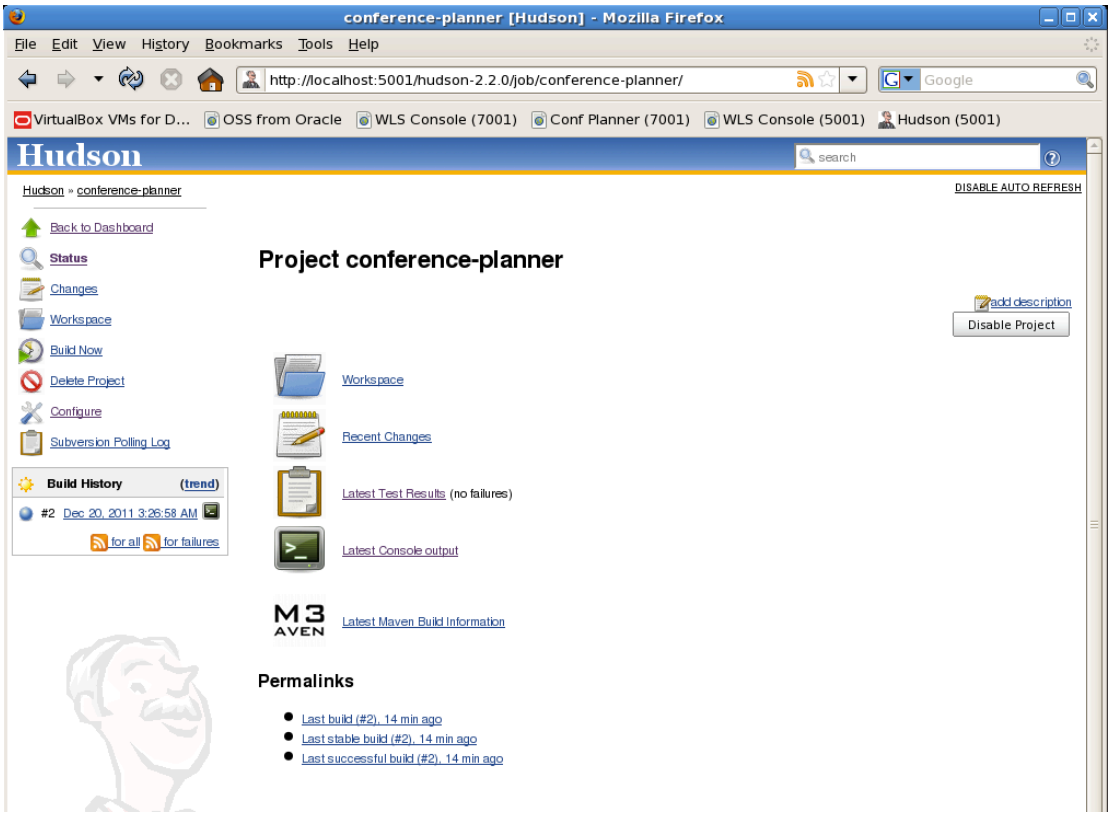


The dashboard will show the result of the job. If the job has completed successfully it will be reflected by a bright shining start. Move the cursor over

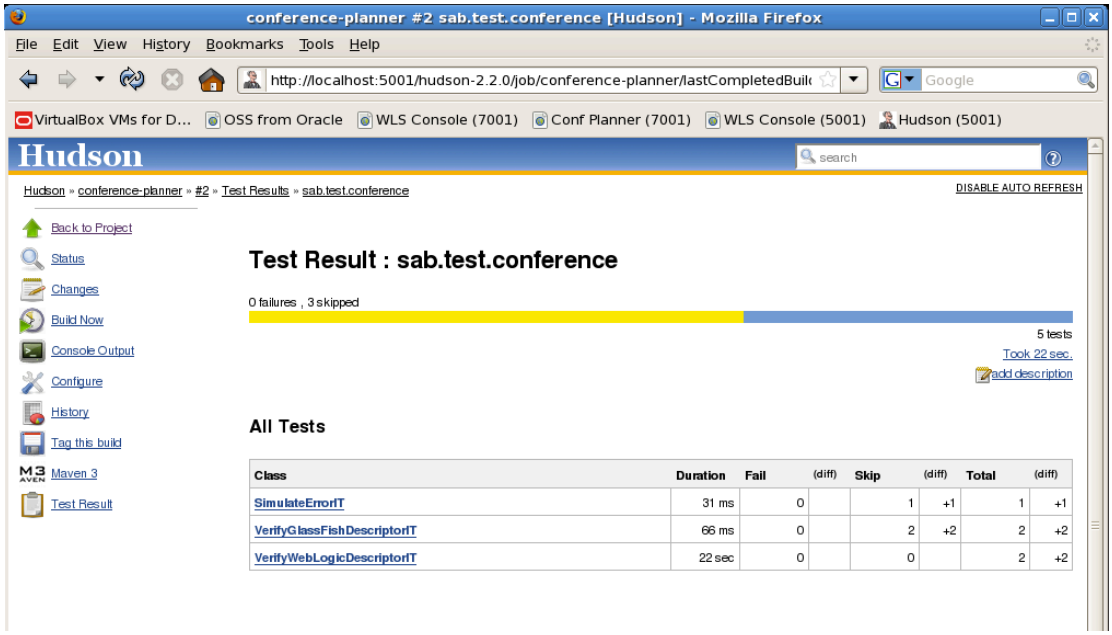
the different icons in the status panel to see a summary of the recent status of the job.



Click the **Conference Planner** link to drill down into the details of the latest Conference Planner job. From here you can drill down further into the details of the job including the contents of the workspace, the changes that were part of the latest SCM update of the project and the console output again.



Click the **Latest Test Results** link to view the detailed results of the tests that were run as part of the project execution.



The test summary shows there are three test cases that were detected in the project and that the VerifyWebLogicDescriptorIT test case was executed without any issues. The VerifyGlassFishDescriptorIT test case contained two skipped tests.

Note: the VerifyWebLogicDescriptorIT test case performs two tests on the ConferencePlanner application, expecting the application to be accessible via the context-root binding specified in the weblogic.xml file: /ConferencePlanner.

```
public class VerifyWebLogicDescriptorIT {

    // This is the URL when the application is deployed with a
    // weblogic.xml file, for which the context-root is
    // ConferencePlanner

    String URL = "http://localhost:7001/ConferencePlanner/test";

    @Test
    public void testDevTestServlet() throws Exception {
        final WebClient webClient = new WebClient();
        final HtmlPage page = webClient.getPage(URL);
        assertEquals("DevTestServlet", page.getTitleText());
        webClient.closeAllWindows();
    }

    @Test
    public void testDevTestServletDataAccess() throws Exception {
        final WebClient webClient = new WebClient();
        final HtmlPage page = webClient.getPage(URL);
        final String pageAsText = page.asText();
        assertTrue(pageAsText.contains("Total Sessions: 432"));
        assertTrue(pageAsText.contains("Total Speakers: 501"));
```

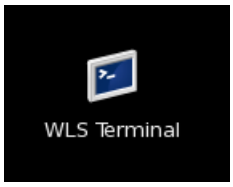
```
        webClient.closeAllWindows();
    }
```

After you have finished exploring, click the Hudson link at the top of the page to go back to the dashboard view.

5. Make and change to the project and check it in

To observe how Hudson continuously watches and monitors the quality of a project, you will now make a change to the project to alter it slightly and introduce a test failure, then observe how Hudson reacts and reports the change.

Open the WLS terminal again.



In the terminal, use VI to edit the pom.xml in the WLS-Maven project you used in part 2.

```
$ cd ~/labs/Dev_labs_2012/WLS-Maven
$ cd conference-planner/trunk
$ vi pom.xml
```

In the pom.xml file, locate the maven-war-plugin definition and **uncomment** the following section to exclude the weblogic.xml deployment descriptor from the war file when it is packaged.

<!—
Uncomment this to see the glassfish-web.xml descriptor support in action, by excluding weblogic.xml
-->
<!--
<packagingExcludes>WEB-INF/weblogic.xml</packagingExcludes>
-->

By excluding the weblogic.xml descriptor, the war file will only contain the glassfish-web.xml descriptor. The glassfish-web.xml descriptor sets the context-root for the web module to /ConferencePlanner_GF.

As WebLogic Server 12*c* supports the reading and application of a set of values from GlassFish deployment descriptors when no competing WebLogic descriptor is provided, this packaged war file will now be mapped to the GlassFish deployment descriptor context-root value of ConferencePlanner_GF.

This will result in the VerifyWebLogicDescriptorIT test case failing since it uses the context-root specified in the weblogic.xml file, which is no longer used.

Once you have made the change to the pom.xml file, check it back into subversion using the commit command.

```
$ svn commit –m "use gf descriptor"

Sending        trunk/pom.xml
Transmitting file data .
Committed revision 75.
```

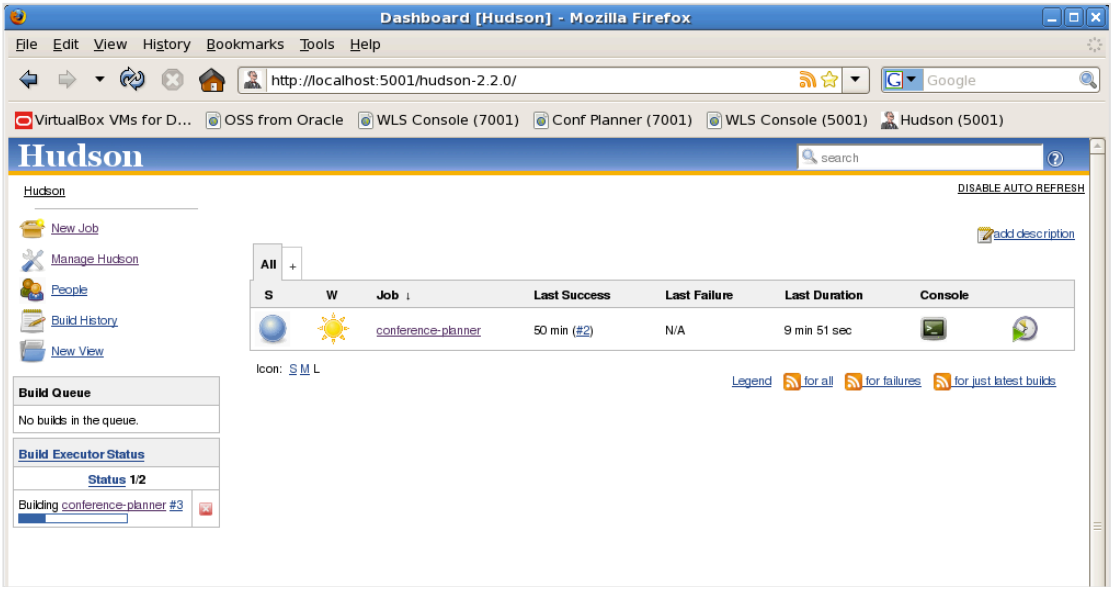
This commits the change back into the subversion repository.

On the commit being completed, subversion directs Hudson using a HTTP request to poll the conference-planner project for the changes.

6. Observe results on Hudson

Go back to the Hudson dashboard in the browser window.

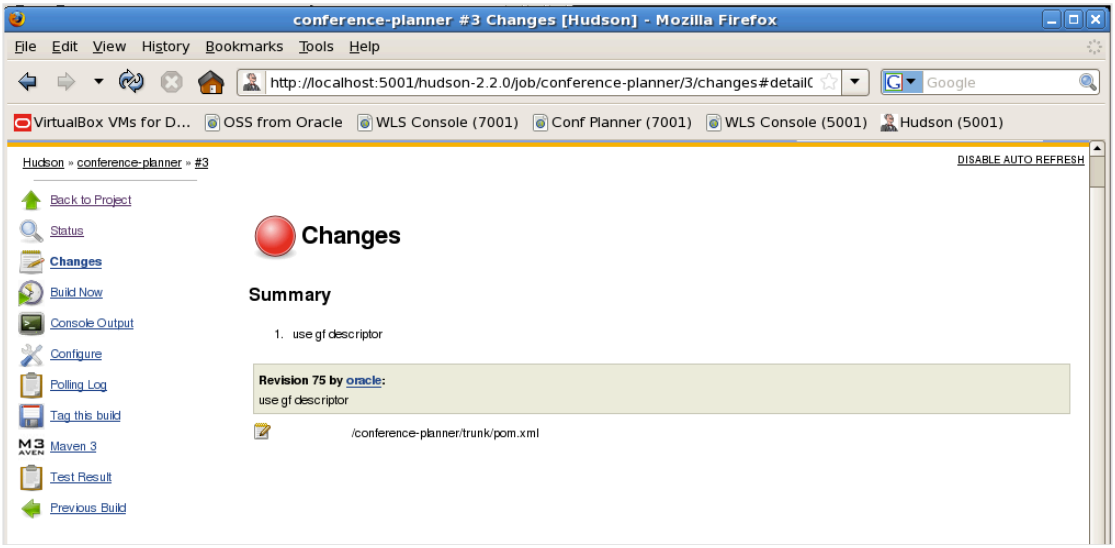
From the dashboard view you will see a new job executing in the **Build History** portlet.



Click the **currently active build** to drill down into the project being built.

To see the changes in the project based on this job, click the **Changes** link. You will see the check in message you specified when you performed the svn commit operation.

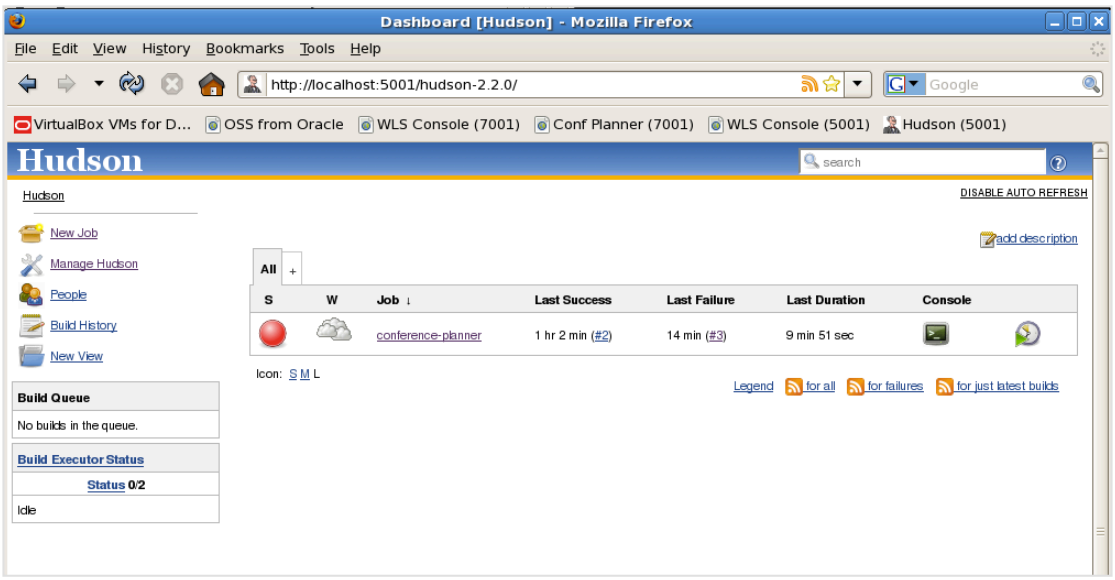
To see the files changed with this commit, click the **detail** link.



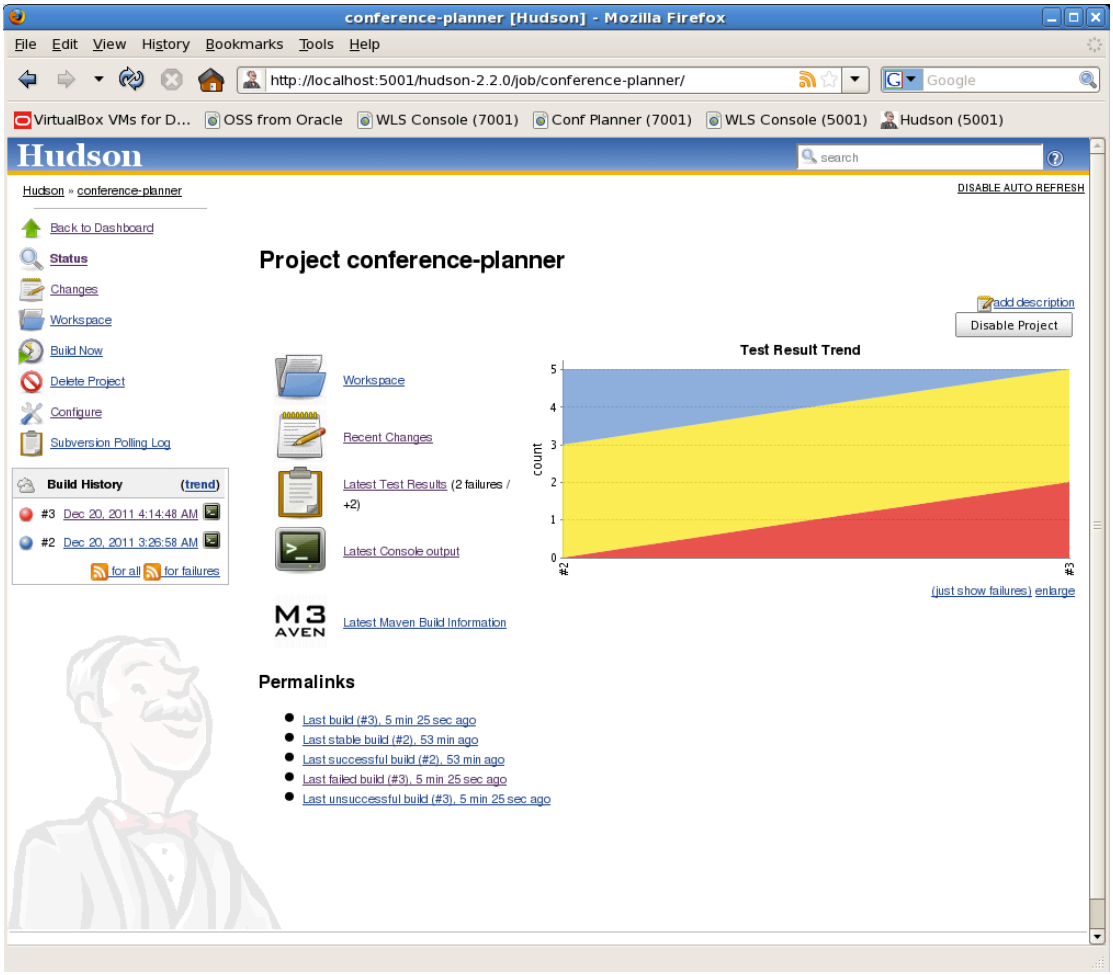
After a short while the job should complete.

Click the **Hudson** link at the top of the page to see the dashboard view.

The conference-planner job will now show that a failure has occurred.

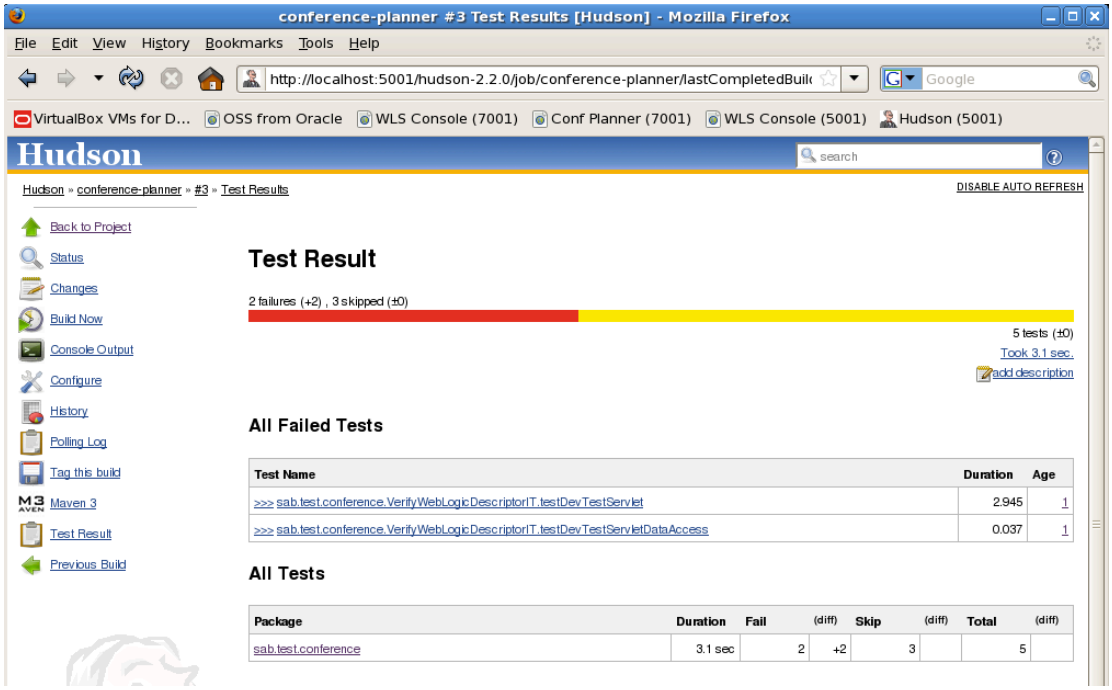


Click the **conference-planner** link to drill down into the job to find out more about what failed.

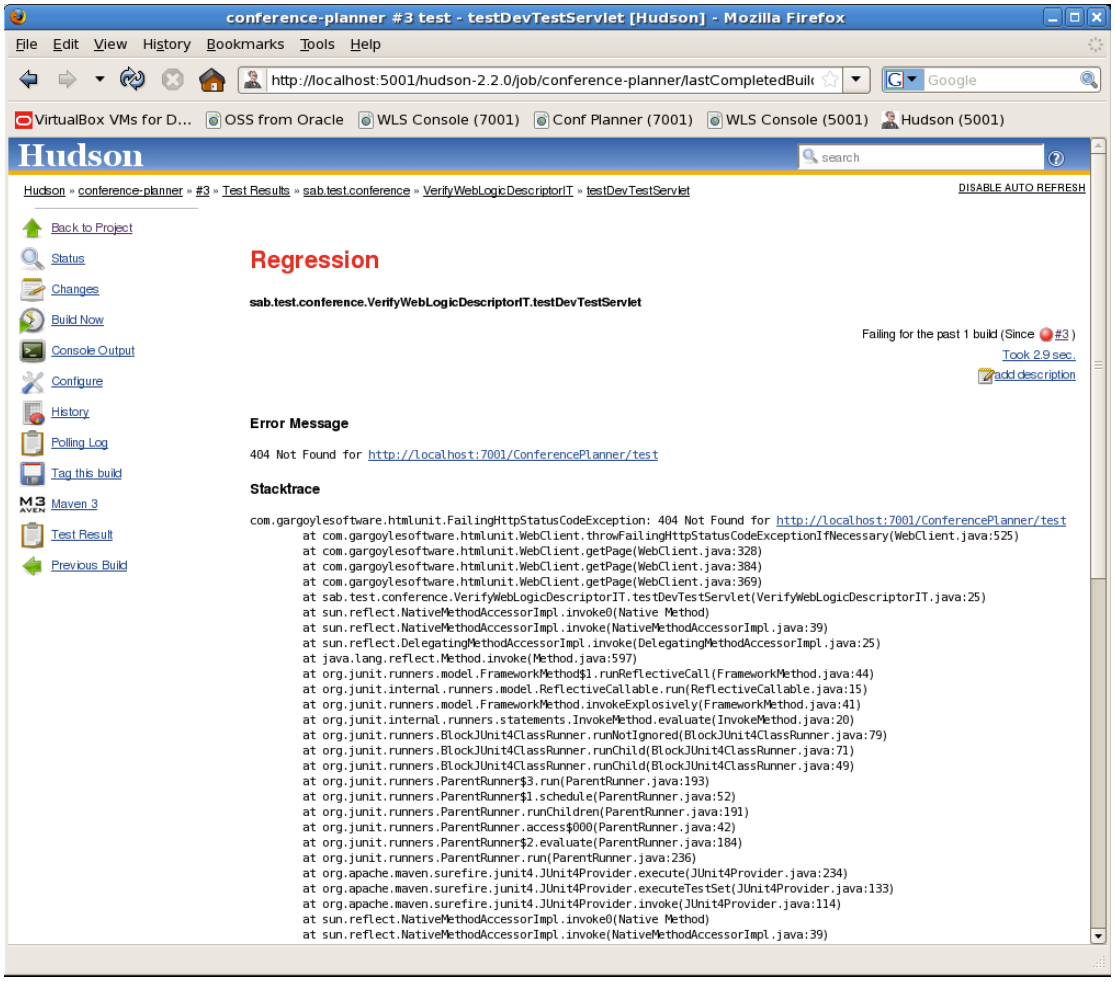


The **Latest Test Results** value shows there are now 2 failures, and that is an increase of 2 from the last build.

Click the **Latest Test Results** link to drill down into the test results to identify the tests that failed.



Drilling down further into the **VerifyWebLogicDescriptorIT** » **testDevTestServlet** test method, the details show that the test fails with a HTTP 404 error.



This verifies that the prior weblogic.xml context-root value is no longer being used after the change to exclude weblogic.xml from the war file was committed to the subversion repository.

To positively verify that the glassfish-web.xml descriptor is being used, it is left as an exercise for the interested reader to seek out the source of the **VerifyGlassFishDescriptorIT** test, remove the @Ignore annotation from the test methods, commit the change and verify the results in Hudson.

Bonus task: can you then produce a fully clean test run with the GlassFish descriptor being used?

If you'd like to re-run the exercise, just do a little cleanup:

1. clean up wls

in turn, start each WLS domain and undeploy any applications, libraries, etc. leave any datasources, then stop it

2. remove the hudson workspace

$ cd ~

$ rm -fr ~/.hudson

3. clean up lab working directory

$ rm -fr ~/labs/Dev_labs_2012/WLS-Maven/conference-planner

4. the local maven repo is pre-poplulated but that won't matter since it overwrites cleanly. we can clean it if you want, but i didn't bother.

$ cd ~/.m2/repository/com/oracle/weblogic/
$ rm -fr wls-dev
$ rm -fr wls-maven-plugin
$ rm maven-metadata-local.xml

Summary

The Hudson Continuous Integration Server provides a service that enables the quality of a project under active development to be continuously monitored and any regressions quickly discovered and reported.

With Hudson being able to natively work with maven projects, the task of using WebLogic Server as the server under test is greatly simplified with the use of the wls-maven-plugin, which can perform all the required operational and deployment related operations directly from the maven execution environment.

Part 3 of this hands-on lab demonstrated the use of the wls-maven-plugin with the Conference Planner project, enabling a completely automated testing environment to be created.

Summary

Describe overall hands-on lab summary.